ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΕΝΤΡΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΤΕ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Τεχνολογία powerline homeplug για την δημιουργία του «έξυπνου» σπιτιού και εφαρμογή της στην κατασκευή αλληλεπιδραστικής διεπαφής ελέγχου και χειρισμού συσκευών στο «έξυπνο» σπίτι, χρησιμοποιώντας οπτικό προγραμματισμό σε C++.

Γεώργιος Δ. Μήτκας

Επιβλέπων : Δρ Ευάγγελος Δημητριάδης
Επιστημονικός Συνεργάτης

ΣΕΡΡΕΣ

2016

«Δηλώνω υπεύθυνα πως το παρόν κείμενο αποτελεί προϊόν προσωπικής μελέτης και εργασίας και πως όλες οι πηγές που χρησιμοποιήθηκαν για τη συγγραφή του δηλώνονται σαφώς στη βιβλιογραφία. Γνωρίζω πως η λογοκλοπή αποτελεί σοβαρότατο παράπτωμα και είμαι ενήμερος για την επέλευση των νόμιμων συνεπειών.»

Πίνακας Περιεχομένων

Πίνακας Εικόνων

ΠΕΡΙΛΗΨΗ

Σκοπός της παρούσας πτυχιακής εργασίας είναι η δημιουργία ενός «έξυπνου» σπιτιού, διαχειριζόμενο τοπικά από τον ηλεκτρονικό υπολογιστή , είτε απομακρυσμένα μέσω μηνύματος sms από κινητό τηλέφωνο. Για την υλοποίηση του, θα χρησιμοποιηθεί η τεχνολογία powerline η οποία είναι αρμόδια για την σύνδεση των απαραίτητων συστημάτων καθώς επίσης και η υπολογιστική πλατφόρμα Arduino ως μικροελεγκτής ελέγχου και διαχείρισης της λειτουργίας του.

ABSTRACT

The purpose of this thesis is to create a "smart" house, managed locally by the computer or remotely via sms message from mobile phone. For its implementation, it will be used the powerline technology which is responsible for connecting the necessary systems as well as the computing platform Arduino as microcontroller of control and manage its operations.

Κεφάλαιο 1ο

1.1. Εισαγωγή

Η ραγδαία ανάπτυξη της τεχνολογίας έχει επιφέρει σημαντικές αλλαγές στις ζωές των ανθρώπων. Τέτοιου είδους αλλαγές μπορεί να σχετίζονται με την άνεση και την ασφάλεια που παρέχονται από ένα «έξυπνο» σπίτι. Πολλά πράγματα που πριν δεκαετίες θεωρούνταν αδιανόητα για τον άνθρωπο, σήμερα μπορούν να γίνουν με το πάτημα ενός και μόνο κουμπιού, διευκολύνοντας την καθημερινή ζωή των ανθρώπων και κάνοντας την ασφαλέστερη.

1.2. Ορισμός του «έξυπνου» σπιτιού

Ο όρος αναφέρεται στις δυνατότητες που παρέχει ένα σπίτι για έλεγχο διαφόρων συσκευών – παραμέτρων (π.χ. φωτισμός, ηλεκτρικές συσκευές, σύστημα συναγερμού, σύστημα θέρμανσης- ψύξης κ.α.) από το χρήστη, είτε με τη χρήση ηλεκτρονικών υπολογιστών εγκατεστημένων στον ίδιο χώρο, είτε με τη χρήση κινητής τηλεφωνίας με αποστολή μηνύματος sms. Ένα σπίτι δηλαδή με νοημοσύνη που σκέπτεται και ενεργεί βάση των καθημερινών αναγκών και συνηθειών των ενοίκων του.

1.3. Οι υπηρεσίες του «έξυπνου» σπιτιού

Δύο από τις κυριότερες υπηρεσίες που μπορεί να προσφέρει ένα «έξυπνο» σπίτι είναι η ασφάλεια και η ευκολία διαχείρισης όπως ήδη αναφέραμε. Όσον αφορά τις υπηρεσίες ασφάλειας αξίζει να αναφερθεί η δυνατότητα ειδοποίησης σε περίπτωση παραβίασης του χώρου. Ενώ σχετικά με την ευκολία διαχείρισης, μερικές από τις παρέχουσες υπηρεσίες είναι η δυνατότητα προσαρμογής του φωτισμού υπό διαφορετικές συνθήκες (π. χ ημέρας, νύχτας και διακοπών), η δυνατότητα ρύθμισης του συστήματος ψύξης – θέρμανσης και η δυνατότητα ενεργοποίησης- απενεργοποίησης ποικίλλων ηλεκτρικών συσκευών με απώτερους στόχους, λόγου χάρη την εξοικονόμηση χρόνου για το χρήστη , την εξοικονόμηση ενέργειας κ.α.

1.4. Τεχνολογία powerline homeplug

Η τεχνολογία powerline είναι ένα πρωτόκολλο επικοινωνίας το οποίο χρησιμοποιεί το ήδη υπάρχων ηλεκτρικό δίκτυο του χώρου με σκοπό την μετάδοση δεδομένων στο τοπικό δίκτυο αυτού. Μερικά από τα πρότυπα χρήσης που μπορούν να αναφερθούν είναι το INSTEON, το X10 καθώς επίσης και το Homeplug τα οποία και χρησιμοποιούνται ως επί των πλείστων στον οικιακό έλεγχο. Για την χρήση αυτής της τεχνολογίας απαιτούνται προσαρμογείς δικτύου Powerline κατά ζεύγη όπου ο ένας αναλαμβάνει τον ρόλο του πομπού στο δίκτυο και ο άλλος του δέκτη. Η συχνότητα εκπομπής στο ηλεκτρικό δίκτυο κυμαίνεται ανάμεσα στα 20 και στα 200kHz ανάλογα με τον προσαρμογέα. Ο κάθε δέκτης έχει μια μοναδική διεύθυνση στο δίκτυο και μπορεί να δεχθεί αποκλειστικές εντολές απ' τα σήματα που στέλνει ο πομπός και στην συνέχεια τα αποκωδικοποιεί.

Εικόνα 1.1 : Προσαρμογείς δικτύου τεχνολογίας Powerline Homeplug (Powerline Homeplug network adapters)

1.4.1. Πλεονεκτήματα τεχνολογίας Powerline homeplug

Η τεχνολογία αυτή, προσφέρει αρκετά πλεονεκτήματα, μερικά εκ των οποίων είναι:

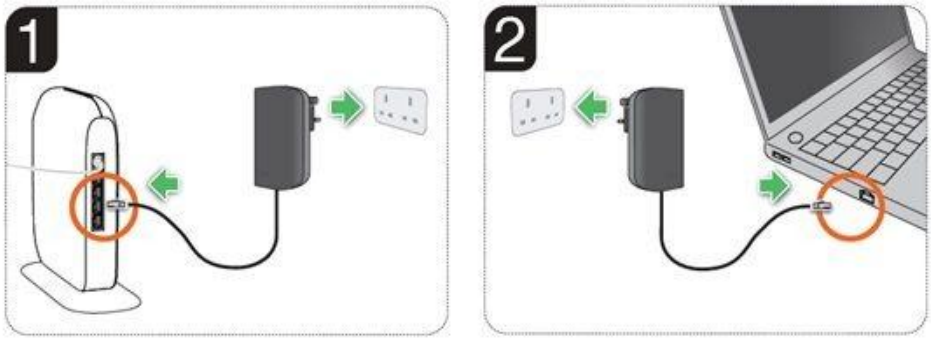
- Εύκολη και άμεση εγκατάσταση των προσαρμογέων, καθώς το μόνο που χρειάζεται είναι η τοποθέτηση τους στις ηλεκτρικές πρίζες, συνδέοντας τον έναν από αυτούς με τον δρομολογητή, και χρησιμοποιώντας τον δεύτερο, ως έξοδο για σύνδεση με συσκευή που θέλουμε να συμπεριλάβουμε στο δίκτυο μας.

- Η χρήση της τεχνολογίας αυτής γίνεται μέσω της υπάρχουσας ηλεκτρικής δικτύωσης, όπως έχει προαναφερθεί, χωρίς να απαιτείται η εγκατάσταση νέων καλωδιώσεων στο σπίτι.
- Η κρυπτογράφηση που χρησιμοποιούν οι προσαρμογείς στην επικοινωνία μεταξύ τους, κάνουν την χρήση τους να φαίνεται τόσο ασφαλή , όσο και η δημιουργία δικτύου χρησιμοποιώντας καλωδίωση Ethernet.

1.4.2. Μειονεκτήματα τεχνολογίας Powerline homeplug

Ωστόσο, σε μερικές περιπτώσεις παρατηρούνται δυσλειτουργίες στην χρήση της, οι οποίες θα μπορούσαν να χαρακτηριστούν ως μειονεκτήματα.

- Σε περιπτώσεις που στην οικία υπάρχουν δύο ή περισσότεροι μετρητές του ηλεκτρικού ρεύματος ή κέντρα ελέγχου , οι δύο προσαρμογείς πιθανόν να μην μπορούν να επικοινωνήσουν μεταξύ τους.
- Το κόστος αγοράς των προσαρμογέων είναι μεγάλο συγκριτικά με την χρήση της καλωδίωσης Ethernet, στις περιπτώσεις που η απόσταση σύνδεσης είναι μικρή.



Εικόνα 1.2 : Παράδειγμα σύνδεσης προσαρμογέων powerline homeplug.

Κεφάλαιο 2ο

2.1. Υπολογιστική πλατφόρμα Arduino

Στο συγκεκριμένο κεφάλαιο θα γίνει αναφορά στα τεχνικά χαρακτηριστικά, τις δυνατότητες , τις λειτουργίες και τις ασπίδες (shields) που αφορούν την υπολογιστική πλατφόρμα Arduino. Στην συνέχεια θα παρουσιαστούν διάφορες εκδόσεις της υπολογιστικής πλατφόρμας και γίνεται επιλογή της έκδοσης που θα χρησιμοποιηθεί στην δημιουργία του «έξυπνου» σπιτιού.

2.2. Ορισμός της υπολογιστικής πλατφόρμας Arduino

Η υπολογιστική πλατφόρμα Arduino είναι μια πλακέτα ανοιχτού υλικού και λογισμικού, η οποία αποτελείται από έναν μικροελεγκτή , από εισόδους / εξόδους είτε ψηφιακές είτε αναλογικές και άλλα εξαρτήματα, με κύριο σκοπό τον έλεγχο ή την διαχείριση αντικειμένων ή συσκευών στον πραγματικό κόσμο, όπως τον χειρισμό του φωτισμού, την ρύθμιση του συστήματος ασφαλείας κ.α.

2.3. Διάφορες εκδόσεις Arduino

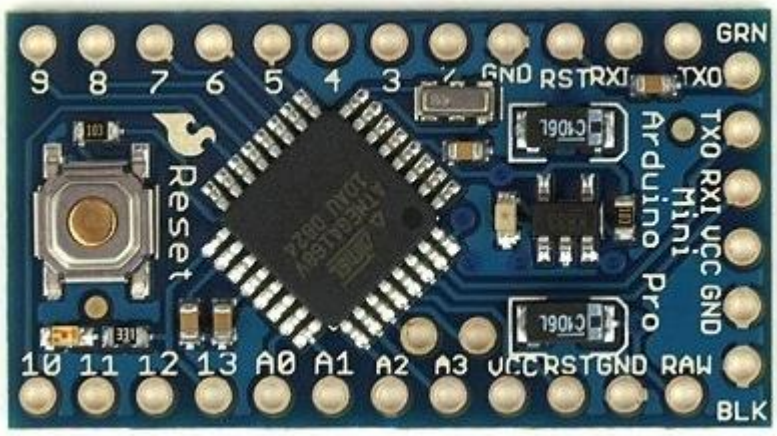
Η παραπάνω υπολογιστική πλατφόρμα έχει κυκλοφορήσει σε αρκετές εκδόσεις με στόχο την κάλυψη των διαφόρων απαιτήσεων που έχουν αναπτυχθεί ανάλογα με το σκοπό λειτουργίας που πρόκειται να εξυπηρετήσουν. Παρακάτω παρουσιάζονται μερικές από αυτές :

2.3.1. Arduino Pro Mini

Η πλατφόρμα Arduino Pro Mini είναι εξοπλισμένη με τα παρακάτω χαρακτηριστικά:

- Μικροελεγκτής ATmega328
- 14 ψηφιακές εισόδους / εξόδους
- 6 αναλογικές εισόδους
- Κουμπί επαναφοράς "reset"
- Υποδοχές τοποθέτησης θυρών οι οποίες δεν προϋπάρχουν

Για την σύνδεση και επικοινωνία με την πλατφόρμα μπορεί να χρησιμοποιηθεί FTDI καλώδιο ή πλακέτα εξόδου της εταιρίας Sparkfun Electronics, η οποία και έχει σχεδιάσει την παραπάνω πλατφόρμα. Τέλος η πλατφόρμα Arduino Pro Mini κυκλοφορεί σε δύο εκδόσεις, μία των 3.3V με συγχρονισμό 8 Mhz, και μια των 5V με συγχρονισμό 16 Mhz, ενώ χρησιμοποιείται συνήθως για προσωρινές εγκαταστάσεις συστημάτων.



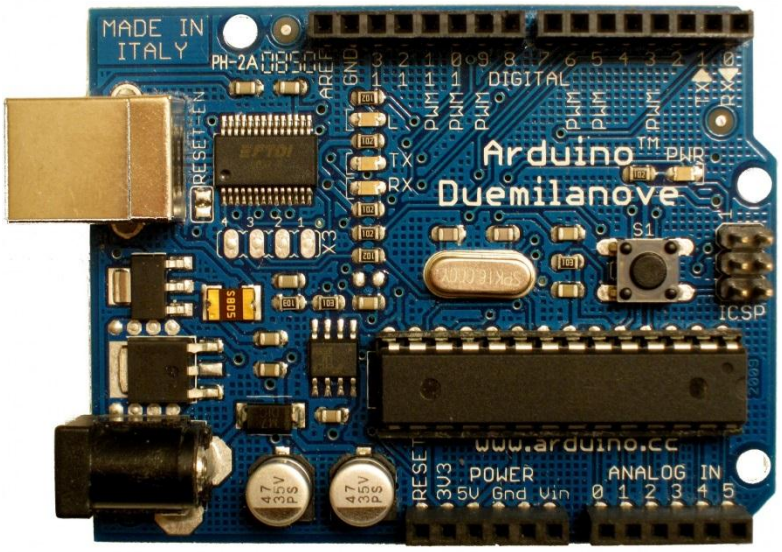
Εικόνα 2.1 : Υπολογιστική πλατφόρμα Arduino Pro Mini.

2.3.2. Arduino Duemilanove

Η πλατφόρμα Arduino Duemilanove παρουσιάζει τα εξής χαρακτηριστικά:

- Μικροελεγκτής ATmega168 ή ATmega328
- 14 ψηφιακές εισόδους / εξόδους
- 6 αναλογικές εισόδους
- 16 MHz ταλαντωτή κρυστάλλου
- Θύρα USB.
- Υποδοχή τροφοδοσίας ρεύματος
- Κουμπί επαναφοράς "reset"

Αξίζει να αναφερθεί πως η παραπάνω πλατφόρμα πήρε το όνομα της από το έτος δημιουργίας της, το οποίο και στα Ιταλικά σημαίνει "2009". Όπως παρατηρούμε μοιάζει αρκετά με την πλατφόρμα Arduino Uno.



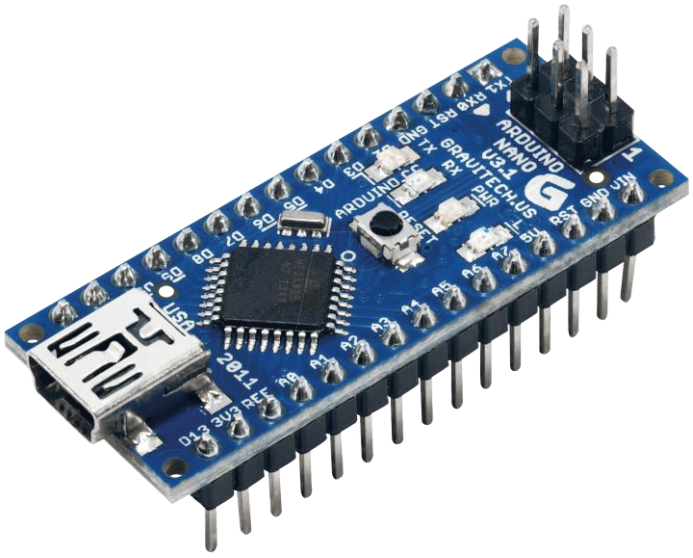
Εικόνα 2.2 : Υπολογιστική πλατφόρμα Arduino Duemilanove.

2.3.3. Arduino Nano

Η πλατφόρμα Arduino Nano είναι εξοπλισμένη με τα παρακάτω χαρακτηριστικά:

- Μικροελεγκτής ATmega328 στην έκδοση της πλατφόρμας 3.- Μικροελεγκτής ATmega168 στην έκδοση της πλατφόρμας 2.-
- Θύρα mini-B USB
- 14 ψηφιακές εισόδους / εξόδους
- 8 αναλογικές εισόδους

Στην ουσία πρόκειται για μια παραπλήσια πλατφόρμα με την Duemilanove, διακρίνοντας τρεις διαφορές, οι οποίες και είναι η έλλειψη τροφοδοσίας ρεύματος σε σχέση με την δεύτερη, η προσθήκη δύο παραπάνω αναλογικών εισόδων και το μέγεθος της.Τέλος αξίζει να αναφερθεί πως σχεδιάστηκε και παράγεται μαζικά από την εταιρία Gravitech.



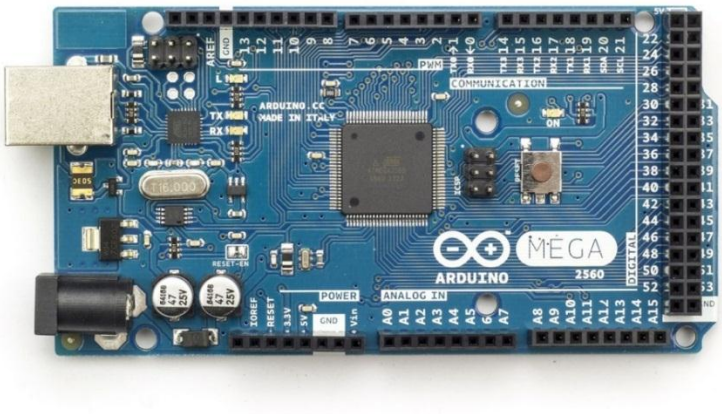
Εικόνα 2.3 : Υπολογιστική πλατφόρμα Arduino Nano.

2.3.4. Arduino Mega 2560

Η πλατφόρμα Arduino Mega έχει τα παρακάτω χαρακτηριστικά:

- Μικροελεγκτής ATmega2560.
- 54 ψηφιακές εισόδους / εξόδους, εκ των οποίων οι 15 μπορούν να χρησιμοποιηθούν ως “ψευδο” αναλογικές, με το σύστημα P.W.M.
- 16 αναλογικές εισόδους.
- 16 MHz ταλαντωτή κρυστάλλου.
- Θύρα USB.
- 4 σειριακές θύρες UARTS.
- Υποδοχή τροφοδοσίας ρεύματος.
- Κουμπί επαναφοράς "reset".

Πρόκειται για μια μεγαλύτερη πλατφόρμα σε σχέση με αυτές που ήδη έχουν αναφερθεί, όσον αφορά τις ψηφιακές εισόδους / εξόδους, και τις αναλογικές εισόδους. Θεωρείται κατάλληλη όταν απαιτείται διαχείριση πολλών παραμέτρων.



Εικόνα 2.4 : Υπολογιστική πλατφόρμα Arduino Mega 2560.

2.3.5. Arduino Yun

Η πλατφόρμα Arduino Yun παρουσιάζει τα εξής χαρακτηριστικά:

- Μικροελεγκτής ATmega32u4
- Μικροεπεξεργαστής Atheros AR9331 με υποστήριξη διανομής Linux βασισμένης στο OpenWrt, η οποία ονομάζεται OpenWrt-Yun.
- Υποδοχή θύρας Ethernet και υποστήριξη Wifi.
- Θύρα τύπου A - USB.
- Υποδοχή κάρτας τύπου microSD για αποθήκευση πληροφοριών - δεδομένων.
- 20 ψηφιακές εισόδους / εξόδους εκ των οποίων οι 7 μπορούν να χρησιμοποιηθούν ως «ψευδο» αναλογικοί έξοδοι και 12 ως αναλογικοί είσοδοι.
- 16 MHz ταλαντωτή κρυστάλλου.
- Υποδοχή σύνδεσης micro-USB.
- 3 κουμπιά επαναφοράς "reset".

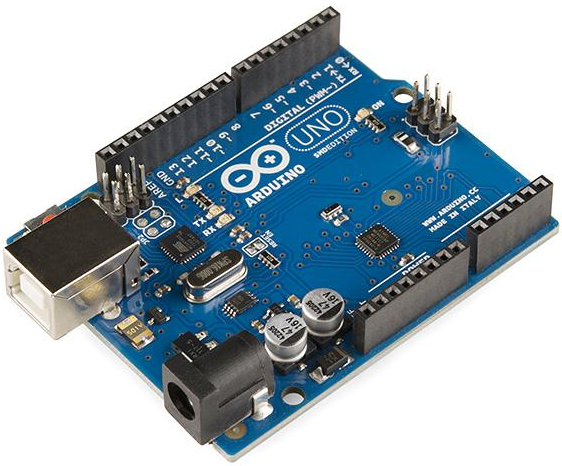
Χρησιμοποιείται κυρίως σε περιπτώσεις που απαιτείται σύνδεση μεταξύ συσκευών, καθώς επίσης και σε περιπτώσεις δημιουργίας συστημάτων της κατηγορίας Internet of Things.

Εικόνα 2.5 : Υπολογιστική πλατφόρμα Arduino Yun.

2.4. Arduino Uno

Η πλατφόρμα Arduino Uno είναι εξοπλισμένη με τα παρακάτω χαρακτηριστικά:

- Μικροελεγκτής ATmega328P,
- 14 ψηφιακές εισόδους / εξόδους με μέγιστη έξοδο ρεύματος 40mA για την κάθε μια.
- 6 αναλογικές εισόδους.
- Θύρα USB για σύνδεση - τροφοδοσία ρεύματος με ηλεκτρονικό υπολογιστή.
- 16 MHz ταλαντωτή κρυστάλλου.
- Υποδοχή τροφοδοσίας ρεύματος για εξωτερική τροφοδοσία.
- Κουμπί επαναφοράς "reset".



Εικόνα 2.6 : Υπολογιστική πλατφόρμα Arduino Uno.

Πρόκειται ουσιαστικά για την πιο διαδεδομένη και ευρέως χρησιμοποιούμενη πλατφόρμα, με αναλυτικές οδηγίες και παραδείγματα για την χρήση της να κυκλοφορούν ελεύθερα στο διαδίκτυο. Για το σκοπό μας, τη δημιουργία του «έξυπνου» σπιτιού, είναι η ιδανική λύση, σε συνδυασμό βέβαια με τις ασπίδες που προαναφέραμε και θα αναλύσουμε στη συνέχεια. Επίσης, παρακάτω αναλύουμε λεπτομερώς τα τεχνικά χαρακτηριστικά του Arduino Uno.

2.4.1. Ο μικροελεγκτής ATmega328

Πρόκειται για έναν 8-bit μικροελεγκτή AVR που παράγεται από την εταιρία Atmel, ο οποίος και βασίζεται στην αρχιτεκτονική RISC. Διαθέτει ενσωματωμένη μνήμη τριών τύπων οι οποίοι παρουσιάζονται παρακάτω.

2.4.2. Η μνήμη "Flash Memory"

Έχει χωρητικότητα 32KBytes εκ των οποίων, τα 512 Bytes είναι δεσμευμένη από το "εσωτερικό πρόγραμμα" που έχει εισάγει ο κατασκευαστής ώστε να μην υπάρχει απαίτηση για εξωτερικό προγραμματισμό υλικού. Το "εσωτερικό πρόγραμμα" αυτό ονομάζεται bootloader. Ο υπόλοιπος χώρος της συγκεκριμένης μνήμης είναι διαθέσιμος για την αποθήκευση του προγράμματος και των δεδομένων που εμείς θα εισάγουμε. Λειτουργεί δηλαδή ως σκληρός δίσκος της πλατφόρμας χωρίς να διαγράφονται τα δεδομένα μετά από αποσύνδεση της τροφοδοσίας ρεύματος ή επαναφοράς αυτής.

2.4.3 Η μνήμη SRAM (Static Random Access Memory)

Μπορεί να θεωρηθεί ως η μνήμη RAM της πλατφόρμας, χωρητικότητας 2KBytes, στην οποία η εφαρμογή που φορτώνουμε, δημιουργεί και επεξεργάζεται μεταβλητές κατά την εκτέλεση της. Αξίζει να σημειωθεί πως στις περιπτώσεις που η εφαρμογή χρησιμοποιεί πολλές μεταβλητές αλφαριθμητικών, υπολογίζοντας πως ένας χαρακτήρας καταλαμβάνει χώρο 1 byte, η μνήμη δεν είναι επαρκής.

2.4.4 Η μνήμη EEPROM

Πρόκειται για μια μνήμη μόνο για ανάγνωση, χωρητικότητας 1024 Bytes, με δυνατότητα αποθήκευσης δεδομένων χωρίς να υπάρχει κίνδυνος απώλειας αυτών λόγω απρόσμενης απενεργοποίησης της πλατφόρμας. Για ανάγνωση και εγγραφή δεδομένων στη συγκεκριμένη μνήμη είναι απαραίτητη η χρήση της ειδικής βιβλιοθήκης και των αντίστοιχων συναρτήσεων που έχουν αναπτυχθεί για το σκοπό αυτό.

2.4.5 Οι αναλογικοί είσοδοι της πλατφόρμας

Η πλατφόρμα Arduino Uno διαθέτει 6 αναλογικές εισόδους με σκοπό χρήσης είτε την συλλογή δεδομένων χρησιμοποιώντας διάφορους αισθητήρες όπως τον αισθητήρα θερμοκρασίας LM35 είτε την εκτέλεση διαφόρων λειτουργιών όπως την ρύθμιση του φωτισμού στο επιθυμητό επίπεδο. Είναι αριθμημένοι χρησιμοποιώντας τους αριθμούς 0 έως 5, αφού πρώτα προηγείται το γράμμα A (Analog). Κατά τον προγραμματισμό του Arduino οι έξοδοι αυτοί μπορούν να αριθμηθούν με τους αριθμούς 14 έως 19. Στις περιπτώσεις που μια αναλογική είσοδο χρησιμοποιείται για συλλογή δεδομένων, γίνεται μετατροπή της τάσεως εισόδου σε έναν 10-bit αριθμό με εύρος τιμής 0 έως 1023.

2.4.6 Οι ψηφιακοί είσοδοι και έξοδοι της πλατφόρμας

Πέρα από τις αναλογικές εισόδους που ήδη περιγράψαμε, η πλατφόρμα μας δίνει την δυνατότητα να εισάγουμε ή να εξάγουμε πληροφορίες, με σκοπό την επεξεργασία ή διαχείριση των παραμέτρων αντίστοιχα. Η πλατφόρμα Arduino Uno διαθέτει 14 ψηφιακές εισόδους / εξόδους οι οποίες πέρα από την απλή λειτουργία τους , μπορούν να χρησιμοποιηθούν ως εξής :

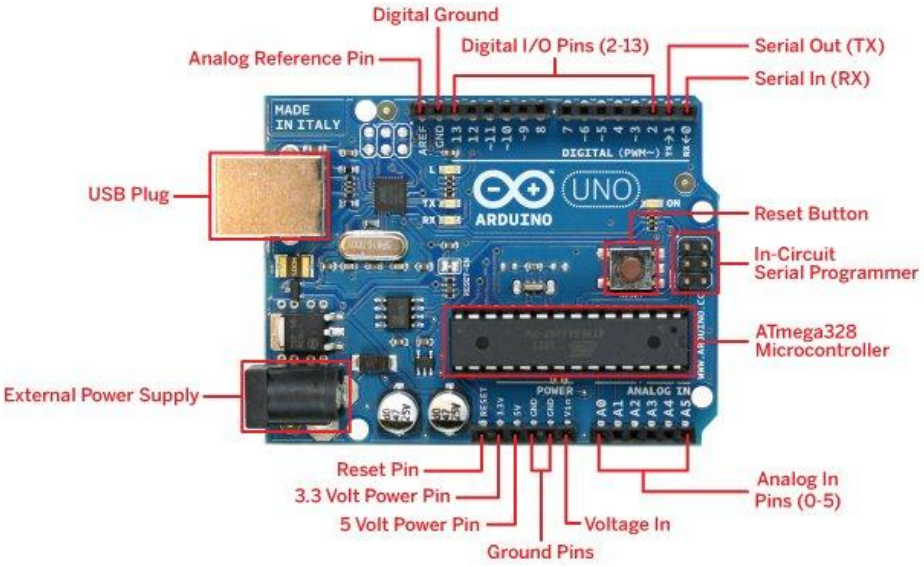
Θέσεις 0 και 1 (RX, TX) : Χρησιμοποιούνται για να λάβουν (receive = RX) ή να μεταδόσουν (transmit = TX) σειριακά δεδομένα. Οι είσοδοι αυτοί, είναι ήδη συνδεδεμένοι με τις αντίστοιχες του τσιπ ATmega8U2 USB - to - TTL. Επί προσθέτως, επιλέγεται η χρήση τους σε περιπτώσεις που θέλουμε να συνδέσουμε καποια επιπλέον συσκευή με την πλατφόρμα , όπως λόγου χάρη μια δεύτερη πλατφόρμα arduino.

Θέσεις 2 και 3 (Εξωτερικά interrupts) : Έχουν τη δυνατότητα να προκαλούν διακοπή στην κανονική ροή του προγράμματος, ανάλογα με τις αλλαγές που δέχονται ως είσοδοι, και να εκτελούν ένα συγκεκριμένο τμήμα δηλαδή μια συνάρτηση. Τα εξωτερικά interrupts είναι πολύ χρήσιμα στη λειτουργία αυτοματοποιημένων συστημάτων καθώς επίσης και στη λύση προβλημάτων στα οποία απαιτείται μεγάλη ακρίβεια.

Θέσεις 3,5,6,9,10 και 11 (PWM outputs) : Πέρα από την λειτουργία ως ψηφιακοί έξοδοι, τα παραπάνω pins μπορούν να χρησιμοποιηθούν και ως "αναλογικοί" έξοδοι χρησιμοποιώντας το σύστημα Pulse Width Modulation, το οποίο και δημιουργεί ομοιόμορφους και χρονικά ίσους επαναλαμβανόμενους παλμούς, διαμορφώνοντας το ψηφιακό σήμα σε "αναλογικό". Το παραπάνω σύστημα μπορεί να δεχτεί ως τιμή από 0 έως 255, χωρίς αυτό να συνεπάγεται πως η ενδιάμεση τιμή θα έχει ως αποτέλεσμα τάση ρεύματος που να ισούται

με 2.5V . Συγκεκριμένα, το σύστημα συνεχίζει να λειτουργεί ψηφιακά (0V ή 5V) δημιουργώντας τους παλμούς που αναφέραμε έτσι ώστε η μέση τιμή τάσης να είναι 2,5V.

Θέσεις 10,11,12,13 : Οι συγκεκριμένες θέσεις υποστηρίζουν την επικοινωνια SPI χρησιμοποιώντας την αντίστοιχη βιβλιοθήκη. Επιπρόσθετα, η θέση 13 είναι ήδη συνδεδεμένη με ένα Led πάνω στη πλατφόρμα, ενεργοποιώντας το όταν η τιμή της θέσης είναι HIGH και απενεργοποιώντας το στην τιμή LOW.



Εικόνα 2.7 : Περιγραφική ανάλυση της υπολογιστικής πλατφόρμας Arduino Uno .

2.4.7 Η τροφοδοσία ρεύματος της πλατφόρμας

Η πλατφόρμα Arduino Uno μπορεί να τροφοδοτηθεί με τρείς διαφορετικούς τρόπους:

- Διαμέσου της θύρας USB.
- Από την παροχή ηλεκτρικού ρεύματος του σπιτιού χρησιμοποιώντας προσαρμογέα σταθερής τάσης με έξοδο βύσματος 2.1mm.
- Από κάποια άλλη πηγή όπως μια μπαταρία χρησιμοποιώντας την γείωση και την θέση Voltage In (Vin) της υπολογιστικής πλατφόρμας.

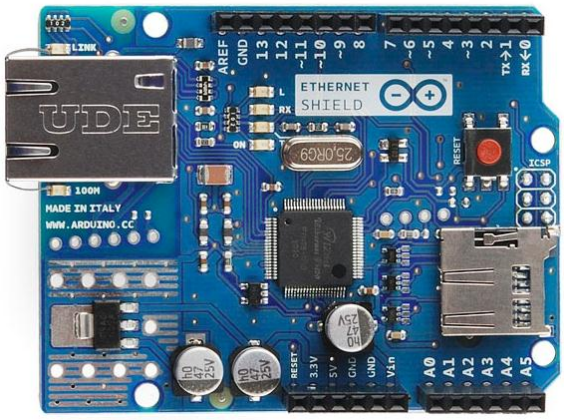
Η πλατφόρμα μπορεί να δεχτεί τάση 6V έως 20V. Αν τροφοδοτηθεί με τάση λιγότερη των 7V ίσως να μην μπορεί να παρέχει 5V στις εξόδους και λόγου αυτού η πλατφόρμα γίνεται ασταθής. Σε περίπτωση τροφοδοσίας μεγαλύτερη των 12V, μπορεί να προκληθεί ζημιά στην πλατφόρμα λόγω υπερθέρμανσης. Γι' αυτούς τους λόγους, καταλήγουμε πως η κατάλληλη τάση μπορεί να κυμαίνεται μεταξύ 7V και 12V, ενώ η προτεινόμενη τάση 9V.

2.5 Ασπίδες της πλατφόρμας (Arduino Shields)

Σε πολλές περιπτώσεις, στην εφαρμογή του συστήματος προς ανάπτυξη, απαιτείται η χρήση εξειδικευμένων εξαρτημάτων - πλακετών, οι οποίες συνδέονται πάνω στην πλατφόρμα Arduino και εξυπηρετούν συγκεκριμένους σκοπούς. Οι πλακέτες αυτές ονομάζονται "ασπίδες" (Shields). Παραδειγματικά αναφερόμαστε στις εξής :

2.5.1 Η ασπίδα Ethernet Shield

Σκοπός της είναι η σύνδεση της πλατφόρμας Arduino, με το οικιακό δίκτυο ή ακόμα και το Internet, χρησιμοποιώντας ένα απλό καλώδιο δικτυού τύπου RJ45 και ακολουθώντας κάποιες οδηγίες που μπορούμε να βρούμε ελεύθερα στο διαδίκτυο σχετικά με τη χρήση της κατάλληλης βιβλιοθήκης. Η τάση λειτουργίας της είναι 5V και παρέχεται από την πλατφόρμα, ενώ η ταχύτητα σύνδεσης είναι 10/100 MBytes. Η ασπίδα συνδέεται συγκεκριμένα με την θύρα SPI του Arduino. Τέλος αξίζει να αναφερθεί πως η συγκεκριμένη πλατφόρμα κυκλοφορεί και σε μια ιδιαίτερη έκδοση, με την υποστήριξη PoE (Power over Ethernet), δηλαδή την τροφοδότηση της μέσω του καλωδίου δικτύου.



Εικόνα 2.8 : Η ασπίδα Arduino Ethernet Shield.

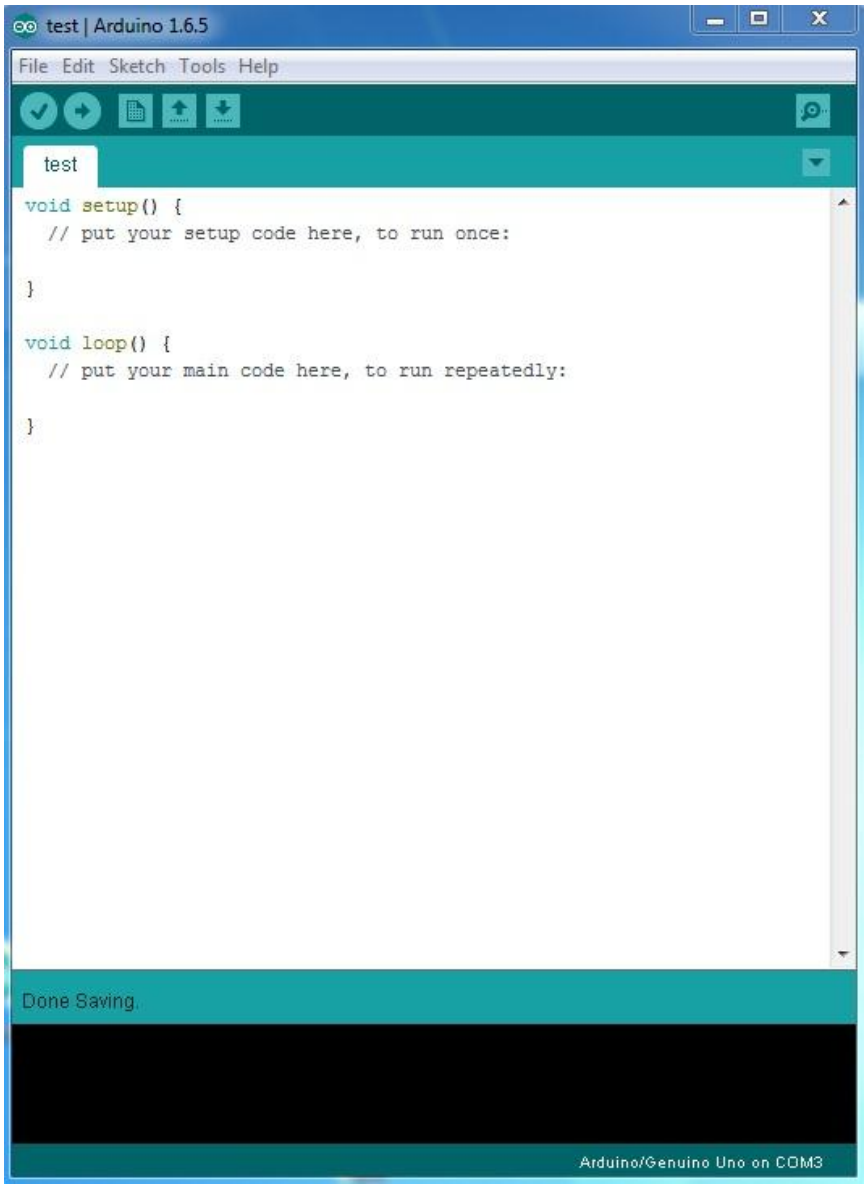
2.5.2 Η ασπίδα GSM Shield

Η ασπίδα αυτή συνδέει την πλατφόρμα Arduino με το Internet χρησιμοποιώντας την υπηρεσία μεταφοράς δεδομένων GPRS, έχει την δυνατότητα να καλέσει ή να δεχθεί κλήσεις (απαιτούνται μικρόφωνο και ακουστικό τα οποία και δεν περιλαμβάνονται) και τέλος να στείλει ή να παραλάβει μηνύματα SMS. Η ασπίδα GSM Shield συνδέεται με το Arduino χρησιμοποιώντας τις θύρες 2,3 και 7, ενώ τροφοδοτείται με 5V από αυτό. Τέλος, περιλαμβάνεται υποδοχή κάρτας SIM η οποία και απαιτείται για την εκτέλεση των λειτουργιών που αναφέραμε.

Εικόνα 2.9 : Η ασπίδα Arduino GSM Shield.

2.6 Περιγραφή Arduino IDE

Όπως έχουμε ήδη αναφέρει, η υπολογιστική πλατφόρμα Arduino είναι υπεύθυνη για τον έλεγχο και την διαχείριση συσκευών σε πραγματικό χρόνο. Για τον σκοπό αυτό, είναι απαραίτητο να γίνει προγραμματισμός της πλατφόρμας ώστε η συμπεριφορά της να είναι ανάλογη με τις εντολές που δέχεται από το χρήστη. Η κοινότητα του Arduino έχει δημιουργήσει το δικό της περιβάλλον ανάπτυξης το οποίο και ονομάζεται Arduino IDE. Το συγκεκριμένο περιβάλλον ανάπτυξης θεωρείται πιο απλό και εύχρηστο σε σχέση με διάφορα άλλα όπως το xCode, το Visual Studio ή το Eclipse. Έχει σχεδιαστεί με σκοπό να εισάγει νέους οι οποίοι δεν είναι εξοικειωμένοι με την ανάπτυξη λογισμικού.



Εικόνα 2.10 : Το περιβάλλον ανάπτυξης Arduino IDE.

Το περιβάλλον ανάπτυξης Arduino IDE είναι γραμμένο σε Java και περιλαμβάνει τρία κύρια χαρακτηριστικά:

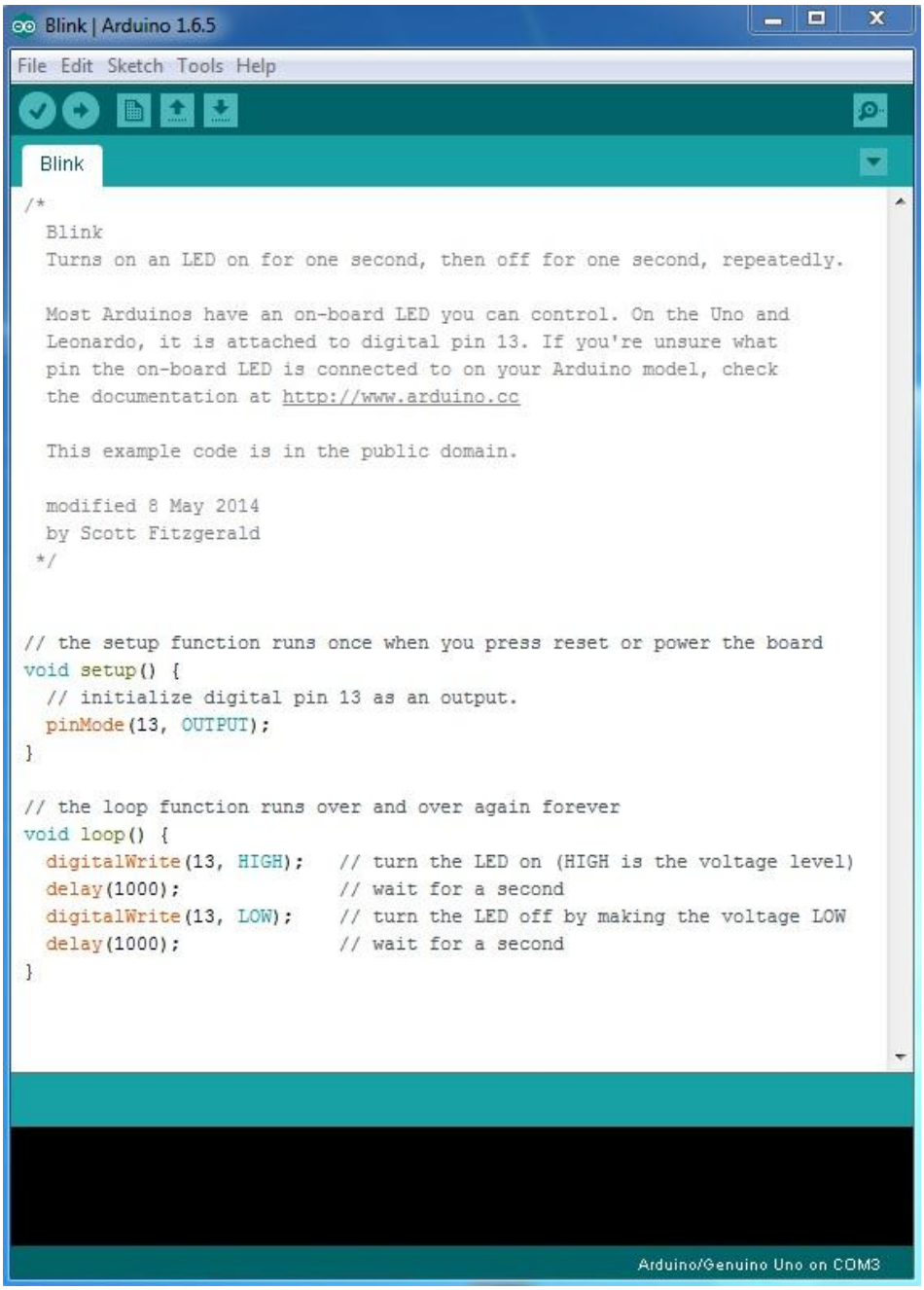
- Πρόγραμμα επεξεργασίας κώδικα (editor) με χαρακτηριστικά όπως η επισήμανση σύνταξης, ο συνδυασμός αγκυλών και η αυτόματη εσοχή

- Μεταγλωττιστή (compiler) ο οποίος μεταγλωττίζει και φορτώνει το κώδικα-πρόγραμμα που έχει ήδη αναπτυχθεί στην πλατφόρμα, μόνο με ένα κλίκ

- Σειριακή οθόνη όπου μπορούν να τυπωθούν διάφορα μηνύματα ενημέρωσης ή ελέγχου κατά την εκτέλεση του κώδικα-προγράμματος

Αξίζει να αναφερθεί πως ένα σημαντικό μειονέκτημα του IDE είναι η έλλειψη της λειτουργίας αποσφαλμάτωσης (debugging) του κώδικα.

Στο Arduino IDE πέρα από τις κλασσικές επιλογές της δημιουργίας , του ανοίγματος, του κλεισίματος και της αποθήκευσης ενός προγράμματος, δίνεται η επιλογή στο χρήστη να επιλέξει ανάμεσα σε μερικά παραδείγματα προγραμμάτων βασικών λειτουργιών που έχουν αναπτυχθεί και συμπεριλαμβάνονται ήδη σε αυτό. Στην παρακάτω φωτογραφία παρατίθεται το παράδειγμα Blink , το οποίο ανοιγοκλείνει ένα led κάθε δευτερόλεπτο.



Εικόνα 2.11 : Το παράδειγμα Blink.

2.6.1 Προγραμματισμός στο Arduino IDE

Κάθε πρόγραμμα που δημιουργείται στο περιβάλλον ανάπτυξης Arduino IDE , ονομάζεται σκίτσο (sketch). Η γλώσσα προγραμματισμού που εφαρμόζεται, ονομάζεται Processing και σε συνδυασμό με την βιβλιοθήκη Wiring γίνεται εφικτός ο προγραμματισμός της υπολογιστικής πλατφόρμας. Το πρόγραμμα χρησιμοποιεί συγκεκριμένες συναρτήσεις που μοιάζουν αρκετά με αυτές της C / C++, καθώς επίσης και συγκεκριμένη δομή. Η δομή των προγραμμάτων στηρίζεται στις δύο βασικές συναρτήσεις **setup()** και **loop()** των οποίων ο ρόλος περιγράφεται παρακάτω.

- **Συνάρτηση setup()** : Μπορεί να θεωρηθεί ως συνάρτηση αρχικοποίησης των τιμών καθώς και προετοιμασίας της εκτέλεσης του βασικού προγράμματος. Η εκτέλεση της γίνεται μόνο μια φορά κατά την τροφοδότηση της πλατφόρμας ή μετά από επανεκκίνηση αυτής.

- **Συνάρτηση loop()** : Πρόκειται για έναν βρόγχο στον οποίο υπάρχει το βασικό μας πρόγραμμα και οι λειτουργίες του , ο οποίος παύει να εκτελείται μόνο μετά από την αποσύνδεσης της πλατφόρμας από το ρεύμα.

2.6.2 Βασικές συναρτήσεις του Arduino IDE

Μερικές από τις βασικές συναρτήσεις που χρησιμοποιήθηκαν για την δημιουργία του προγράμματος για το «έξυπνο» σπίτι και είναι ευρέως γνωστές στον προγραμματισμό του Arduino είναι οι παρακάτω:

- **Serial.begin(speed)** : Ενεργοποιεί την σειριακή θύρα με σκοπό την μεταφορά δεδομένων. Η παράμετρος που δέχεται αφορά τον ρυθμό μεταφοράς , ο οποίος μετριέται σε baud (bits per second).

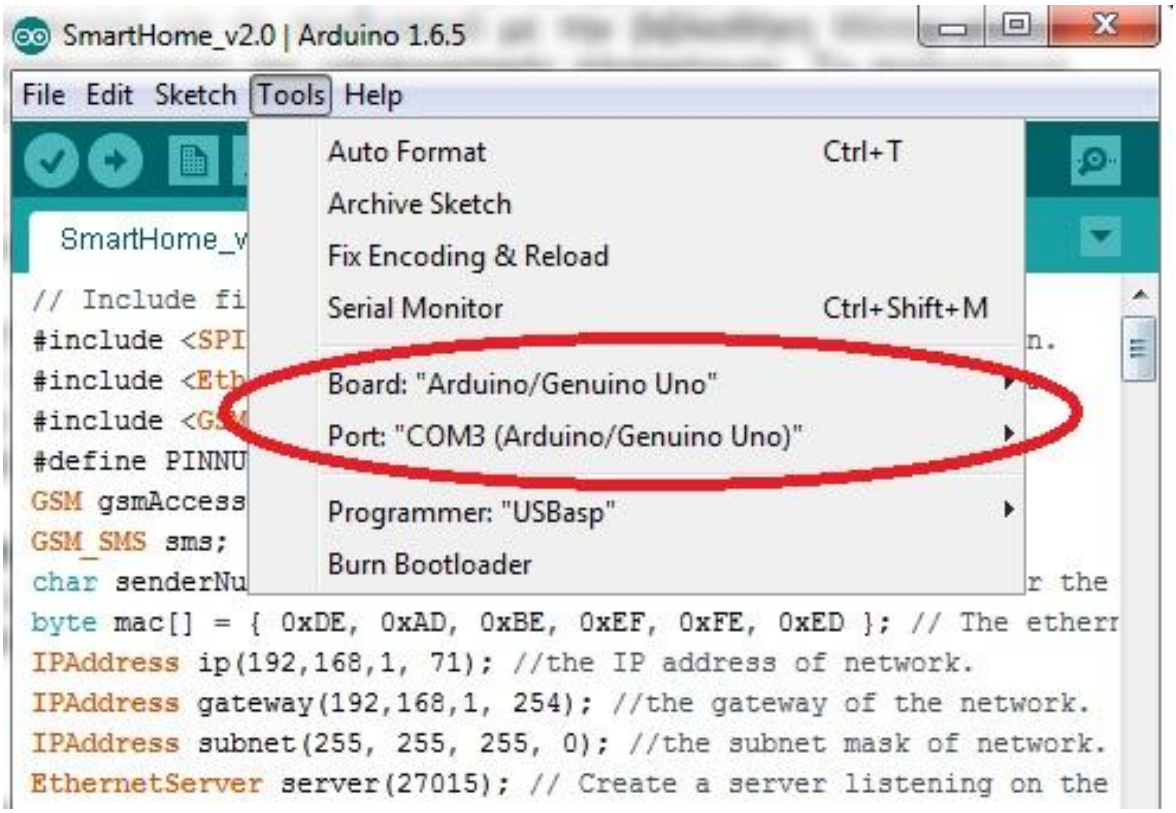
- **pinMode(pin, mode)** : Ορίζει την συμπεριφορά (είσοδος ή έξοδος) για μια συγκεκριμένη ακίδα της πλατφόρμας. Η παράμετρος pin αφορά την τον αριθμό ακίδας που απευθύνεται ο χρήστης ενώ η παράμετρος mode αφορά τον τρόπο συμπεριφοράς.

- **digitalRead(pin)** : «Διαβάζει» την τιμή της ακίδας και μας επιστρέφει την τιμή HIGH ή LOW εάν η ακίδα είναι ενεργοποιημένη η όχι αντίστοιχα. Η παράμετρος pin αφορά τον αριθμό ακίδας που μελετάμε.

- **digitalWrite(pin, value)** : Ενεργοποιεί ή απενεργοποιεί μια συγκεκριμένη ακίδα της πλατφόρμας στέλνοντας την τιμή HIGH ή LOW αντίστοιχα. Σε περίπτωση αποστολής της τιμής HIGH γίνεται τροφοδοσία ρεύματος με τάση 5V ενώ σε αντίθετη περίπτωση η τάση που αποστέλλεται ισούται με 0V.

2.6.3 Σύνδεση υπολογιστικής πλατφόρμας – Arduino IDE

Για την φόρτωση του προγράμματος-σκίτσου είναι απαραίτητη η σύνδεση της υπολογιστικής πλατφόρμας μέσω της σειριακής θύρας με τον υπολογιστή στον οποίο εκτελείται το περιβάλλον ανάπτυξης, καθώς επίσης η επιλογή της πλατφόρμας που έχουμε και της σειριακής θύρας που είναι συνδεδεμένη η πλατφόρμα Arduino Uno, όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 2.12 : Επιλογή έκδοσης πλατφόρμας και σειριακής θύρας.

2.6.4. Βιβλιοθήκες του Arduino IDE

Οι βιβλιοθήκες του Arduino IDE λειτουργούν όπως και οι βιβλιοθήκες που είναι γνωστές στις πλατφόρμες προγραμματισμού. Μερικές από τις λειτουργίες του Arduino θα ήταν ιδιαίτερα περίπλοκες χωρίς την χρήση της κατάλληλης βιβλιοθήκης. Οι βιβλιοθήκες που χρησιμοποιήθηκαν στο έξυπνο σπίτι είναι οι εξής :

- **#include <SPI.h> :** Παρέχει λειτουργίες για την ορθή σύνδεση της υπολογιστικής πλακέτας Arduino με την ασπίδα Ethernet.

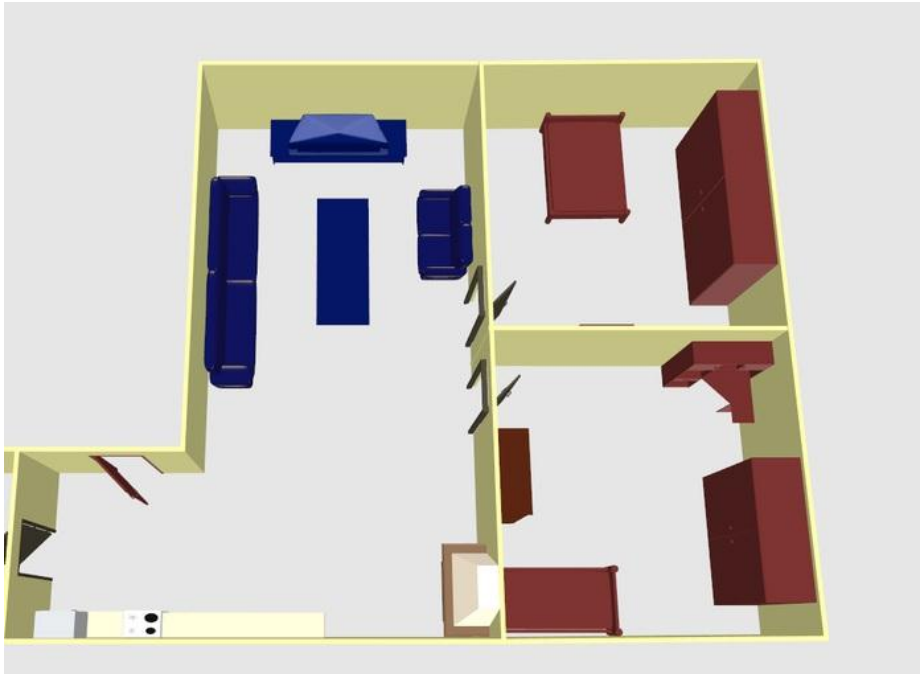
- **#include <Ethernet.h> :** Είναι υπεύθυνη για την σύνδεση και λειτουργία της ασπίδας Ethernet με σκοπό την μετάδοση δεδομένων μεταξύ της υπολογιστικής πλατφόρμας Arduino και του δικτύου.

- **#include <GSM.h> :** Χρησιμοποιείται για την ενεργοποίηση της ασπίδας GSM, με σκοπό την παραλαβή μηνυμάτων SMS από το χρήστη.

Κεφάλαιο 3ο

3.1. Η μακέτα του σπιτιού

Αφού πρώτα μελετήθηκε και ερευνήθηκε η τεχνολογία powerline και η υπολογιστική πλατφόρμα Arduino, κρίθηκε απαραίτητη η δημιουργία μιας μακέτας κατασκευασμένης από ξύλο με σκοπό την επίδειξη και την προσομοίωση του «έξυπνου» σπιτιού. Για την δημιουργία της επιλέχθηκε ξύλο «πλακάζ» ενώ η συναρμολόγηση της πραγματοποιήθηκε χρησιμοποιώντας αδιάβροχη κόλλα ξύλου. Η μακέτα που δημιουργήθηκε χωρίζεται σε 3 χώρους. Το παιδικό δωμάτιο , το υπνοδωμάτιο και έναν ενιαίο χώρο με χρήση σαλονιού και κουζίνας. Οι διαστάσεις της είναι 41 x 51 x 20 cm. Παρακάτω παρουσιάζεται το σχέδιο της μακέτας στην εφαρμογή Sweet Home 3D καθώς επίσης και η μακέτα που δημιουργήθηκε.



Εικόνα 3.1 : Σχέδιο μακέτας

Εικόνα 3.2 : Υλοποιημένη μακέτα

3.2. Μελέτη και αναζήτηση των απαραίτητων εξαρτημάτων

Για τη προσομοίωση του έξυπνου σπιτιού χρησιμοποιήθηκαν τα παρακάτω εξαρτήματα :

- **Arduino Uno Rev 3.** το οποίο και επιλέχθηκε για τους λόγους που ήδη έχουμε αναφέρει. Η συγκεκριμένη υπολογιστική πλατφόρμα αναλαμβάνει ρόλο διαχειριστή και ελεγκτή του «έξυπνου» σπιτιού.

- **Arduino Ethernet Shield**. Σύμφωνα με τις απαιτήσεις και αφού κρίθηκε απαραίτητη η χρήση της τεχνολογίας powerline, είναι απαραίτητη η προσθήκη της συγκεκριμένης πλακέτας με σκοπό την μεταφορά δεδομένων δικτυακά μεταξύ του λογισμικού που αναπτύχθηκε και της πλατφόρμας Arduino. Το πρωτόκολλο επικοινωνίας που χρησιμοποιείτε είναι το TCP/IP, το οποίο είναι ευρέως γνωστό στα δίκτυα υπολογιστών.

- **Arduino GSM Shield**. Με την προσθήκη της συγκεκριμένης πλακέτας δίνεται επιλογή διαχείρισης για τον χρήστη μέσω του τηλεφώνου του, στέλνοντας ειδικά διαμορφωμένα μηνύματα SMS στο προκαθορισμένο αριθμό τηλεφώνου που ανήκει στην κάρτα SIM της ασπίδας.

- **LED 3mm** τα οποία χρησιμοποιήθηκαν για την προσομοίωση των φώτων του σπιτιού και τη τηλεόραση του. Η τάση λειτουργίας των συγκεκριμένων LED είναι μεταξύ 2 και 2.4V σταθερής τάσης ενώ το μέγιστο ρεύμα λειτουργίας ανέρχεται στα 20mA.

Εικόνα 3.3 : LED 3mm

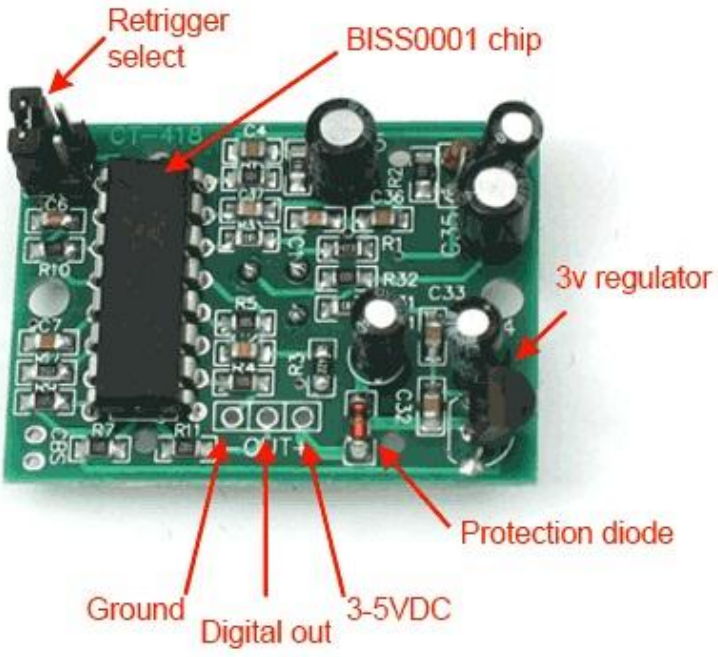
- **Αντιστάσεις 220 Ω** συνδεδεμένες σε σειρά με τα παραπάνω led. Η τιμή της αντίστασης επιλέχθηκε έτσι ώστε το ρεύμα λειτουργίας των led να μην ξεπερνά τα 20mA.

Εικόνα 3.4 : Αντίσταση R = 220 **Ω**

- **Αισθητήρας κίνησης PIR** για το σύστημα συναγερμού του σπιτιού. Λειτουργεί μετρώντας την υπέρυθρη ακτινοβολία που παράγεται από οποιοδήποτε αντικείμενο ή όν του οποίου η θερμοκρασία είναι ανώτερη των 0° C.

Εικόνα 3.5 : Αισθητήρας κίνησης PIR

Η τάση λειτουργίας του μπορεί να είναι μεταξύ των 3V και 5V σταθερής τάσης και τροφοδοτείται από την πλατφόρμα Arduino. Όταν αντιλαμβάνεται κίνηση στέλνει σήμα εξόδου με τιμή HIGH ενώ σε αντίθετη περίπτωση η τιμή του παραμένει στο LOW. Παρακάτω παρουσιάζεται αναλυτικό σχεδιάγραμμα του.



Εικόνα 3.6 : Αναλυτικό σχεδιάγραμμα αισθητήρα PIR

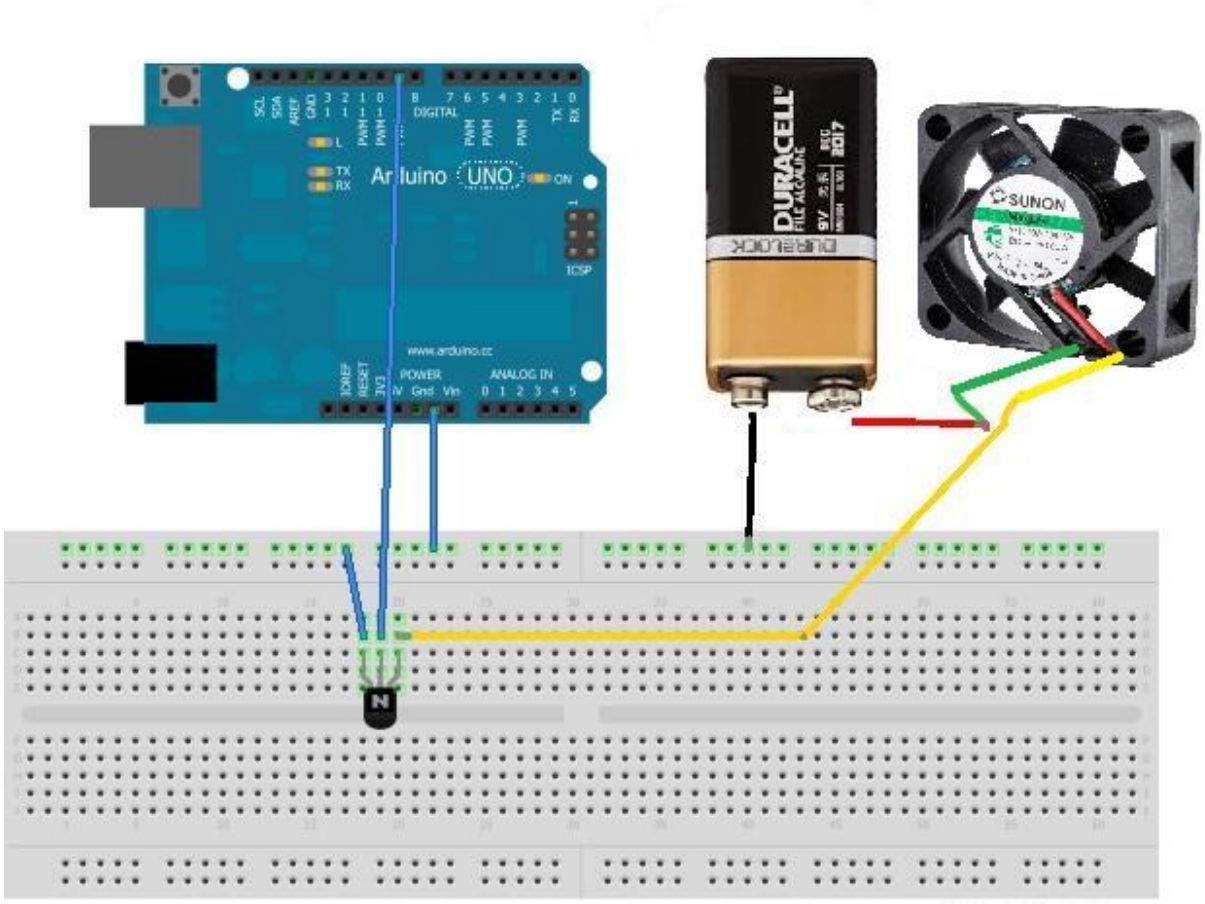
- **Ηχείο** σε περίπτωση παραβίασης του συναγερμού με τάση λειτουργίας τα 6V , ενώ δουλεύει άψογα και με τα 5V που παρέχονται από το Arduino.

Εικόνα 3.7 : Ηχείο συναγερμού.

- Ανεμιστήρας το οποίο προσομοιώνει το σύστημα ψύξης του σπιτιού. Η τάση λειτουργίας του είναι 5V σταθερής τάσης με ισχύ 0.9W.

Εικόνα 3.8 : Ανεμιστήρας – Σύστημα Ψύξης

Σύμφωνα με τον νόμο του Joule $P = V \cdot I$ παρατηρούμε πως το I = 180mA ενώ όπως έχει ήδη αναφερθεί το μέγιστο ρεύμα που μπορεί να παρέχει η κάθε έξοδος του Arduino είναι 40mA. Λόγω ανεπαρκούς τάσης ρεύματος λοιπόν, χρησιμοποιήθηκε μια εξωτερική πηγή ρεύματος, δηλαδή μια μπαταρία σταθερής τάσης των 9V και ένα τρανζίστορ ως διακόπτης ενεργοποίησης / απενεργοποίησης. Στην παρακάτω εικόνα παρουσιάζεται η συνδεσμολογία του κυκλώματος.

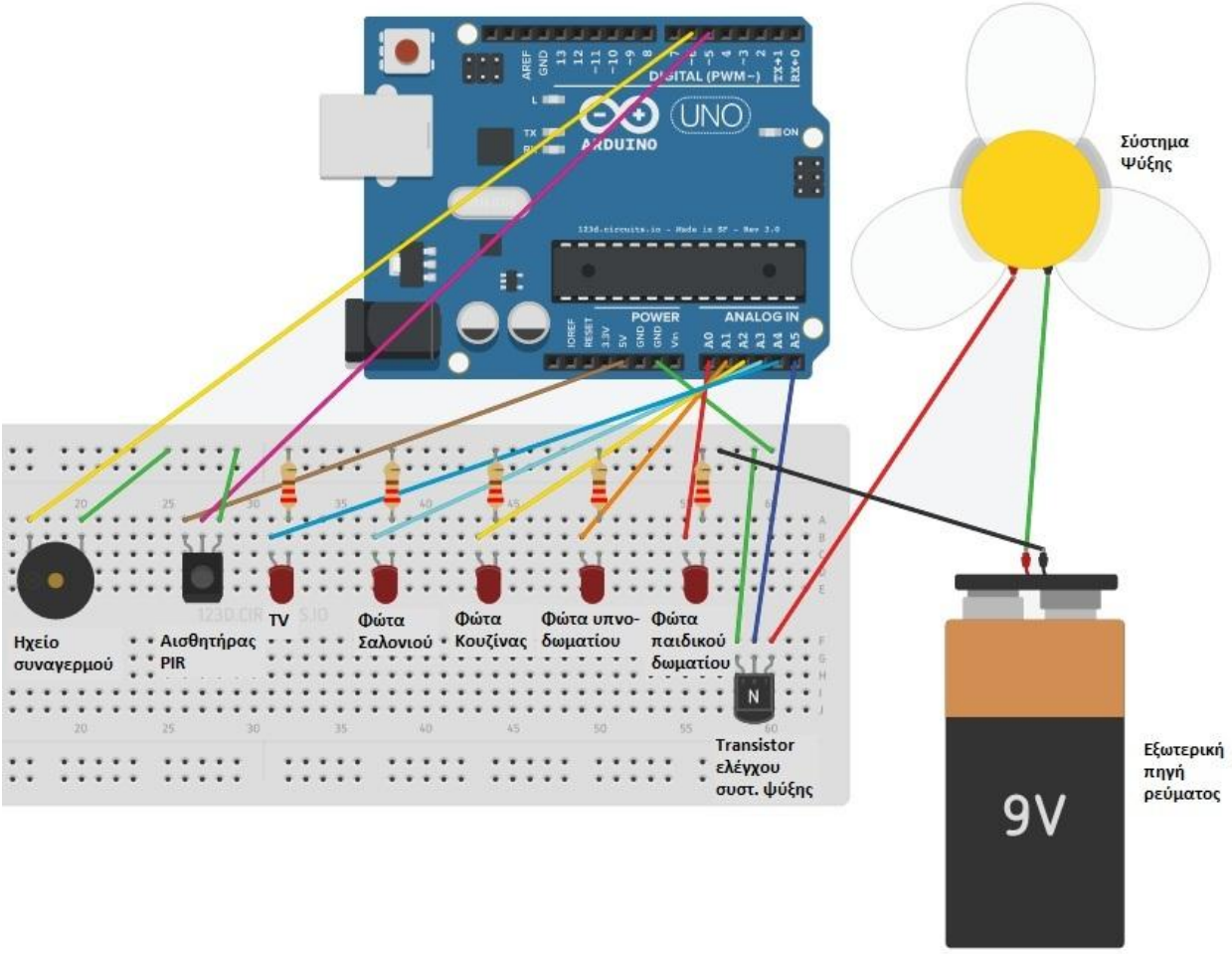


Εικόνα 3.9 : Κύκλωμα Transistor – Σύστημα Ψύξης.

- Καλώδια επέκτασης και πινέζες στήριξης των καλωδίων.

3.3. Συγκέντρωση και συναρμολόγηση των εξαρτημάτων στην μακέτα

Μετά την επί μέρους παρουσίαση των εξαρτημάτων που χρησιμοποιήθηκαν, τοποθετήθηκαν στην μακέτα με τρόπο κατάλληλο για την προσομοίωση του «έξυπνου» σπιτιού. Όλα τα εξαρτήματα είναι συνδεδεμένα σε σειρά με την υπολογιστική πλατφόρμα ενώ στην παρακάτω εικόνα απεικονίζεται η συνδεσμολογία τους όπως αυτή εφαρμόστηκε.

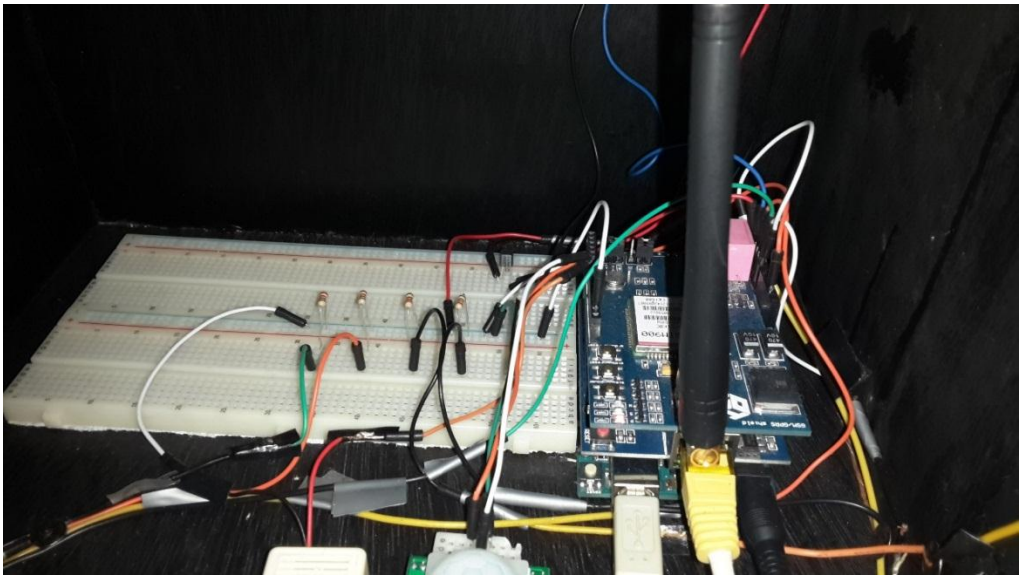


Εικόνα 3.10 : Συνδεσμολογία εξαρτημάτων.

Αξίζει να αναφερθεί πως οι ασπίδες Ethernet και GSM συνδέθηκαν απευθείας επάνω στην υπολογιστική πλατφόρμα Arduino Uno. Στην εικόνα που ακολουθεί ξεκινώντας από πάνω προς τα κάτω βλέπουμε τις ασπίδες GSM και Ethernet, ενώ στο κάτω μέρος έχουμε την πλατφόρμα Arduino.

Εικόνα 3.11 : Σύνδεση Arduino με Ethernet και GSM Shields.

Για την πλήρη εφαρμογή της συνδεσμολογίας χρησιμοποιήθηκαν καλώδια χαλκού ως προέκταση μεταξύ των ακίδων της πλατφόρμας και των εισόδων – εξόδων του κάθε εξαρτήματος. Το αποτέλεσμα της συνδεσμολογίας στην μακέτα φαίνεται στην παρακάτω εικόνα.


Εικόνα 3.12 : Συνδεσμολογία στη μακέτα.

3.4. Λειτουργίες του «έξυπνου» σπιτιού

Ως εφαρμογή, το «έξυπνο» σπίτι μπορεί να διαχειριστεί οποιαδήποτε συσκευή ηλεκτρικού ρεύματος. Ενδεικτικά οι λειτουργίες που επιλέχθηκαν να αναπτυχθούν στην προσομοίωση είναι οι εξής :

- Ενεργοποίηση / Απενεργοποίηση των φώτων της κουζίνας, του σαλονιού, του υπνοδωματίου και του παιδικού δωματίου.

- Ενεργοποίηση / Απενεργοποίηση της τηλεόρασης η οποία και βρίσκεται στο σαλόνι του σπιτιού.

- Ενεργοποίηση / Απενεργοποίηση του συστήματος ψύξης.

- Ενεργοποίηση / Απενεργοποίηση του συστήματος συναγερμού που έχει εγκατασταθεί.

Οι παραπάνω λειτουργίες είναι διαθέσιμες για τον χρήστη είτε μέσω της εφαρμογής του υπολογιστή, είτε μέσω αποστολής μηνύματος SMS από το κινητό του τηλέφωνο.

3.5. Καταστάσεις λειτουργίας του «έξυπνου» σπιτιού

Εκτός από τις λειτουργίες που ήδη αναφέραμε, υπάρχει επιλογή στο χρήστη να ορίζει την κατάσταση λειτουργίας του σπιτιού. Ως κατάσταση λειτουργίας ορίζεται η επιλογή του τρόπου συμπεριφοράς του σπιτιού από το χρήστη. Πρόκειται ουσιαστικά για επιλογή μεταξύ αυτοματοποιημένων λειτουργιών. Παρακάτω παρουσιάζονται οι τρεις καταστάσεις που δημιουργήθηκαν :

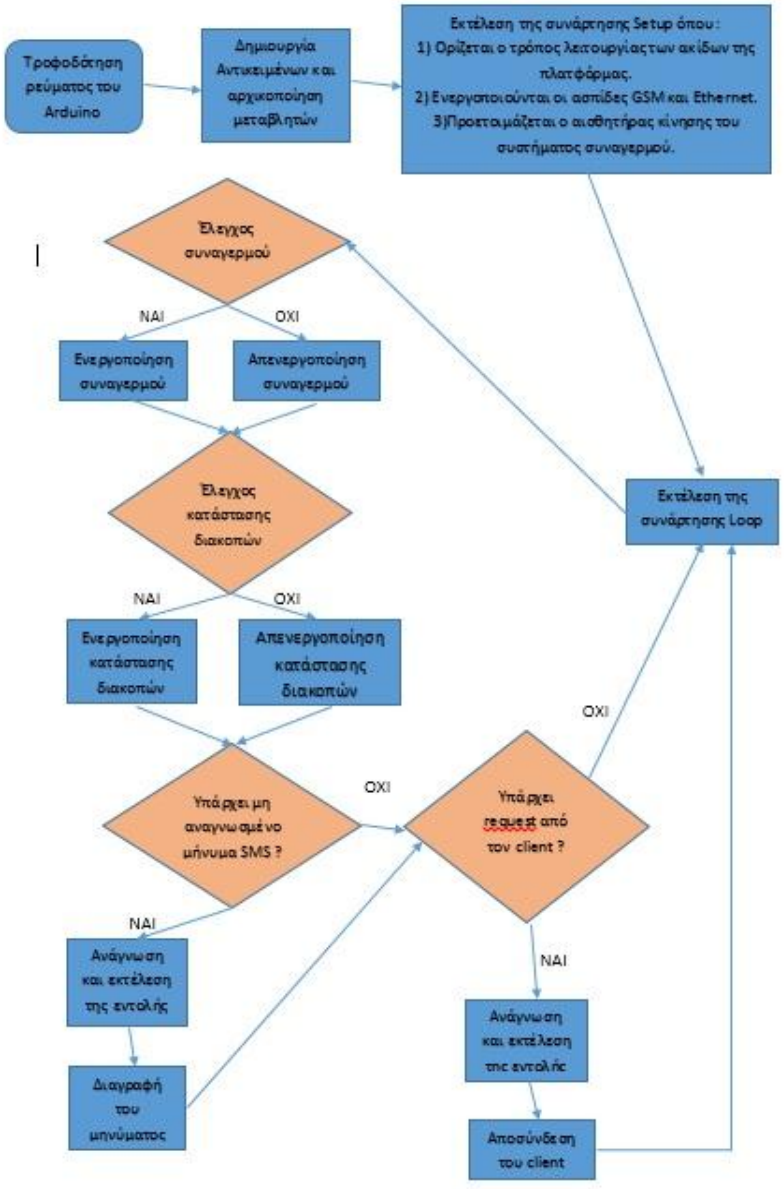
- **Κατάσταση Ημέρας** : Είναι η προεπιλεγμένη κατάσταση κατά την εκτέλεση της εφαρμογής του υπολογιστή. Ο χρήστης κατά τη διάρκεια της ημέρας είναι αυτός που ορίζει τις λειτουργίες του έξυπνου σπιτιού.

- **Κατάσταση Νύχτας :** Κατά την ενεργοποίηση της, το «έξυπνο» σπίτι απενεργοποιεί τη λειτουργία της τηλεόρασης καθώς και όλων των φώτων του σπιτιού εκτός από αυτό της κουζίνας. Στη συνέχεια, ενεργοποιεί το σύστημα συναγερμού έως ότου ο χρήστης αλλάξει την κατάσταση λειτουργίας.

- **Κατάσταση Διακοπών :** Κατά την ενεργοποίηση της, απενεργοποιούνται όλα τα φώτα του σπιτιού και οποιαδήποτε άλλη ηλεκτρική συσκευή είναι σε λειτουργία. Το σύστημα συναγερμού τίθεται σε κατάσταση λειτουργίας. Τέλος, για την αποτροπή πιθανής παραβίασης, ενεργοποιείται αυτόματα το φως της κουζίνας ανά διαστήματα που έχουν οριστεί με χρονικό διάστημα λειτουργίας επίσης ορισμένο. Στη προσομοίωση, το φως λειτουργεί για πέντε δευτερόλεπτα, ενώ παραμένει απενεργοποιημένο για τα επόμενα δέκα.

3.6. Το σκίτσο του «έξυπνου» σπιτιού

Για την εκτέλεση των λειτουργιών και καταστάσεων που αναφέραμε, αναπτύχθηκε ένα «σκίτσο» (sketch) στο Arduino. Το παρακάτω διάγραμμα εξηγεί τις ενέργειες που επαναλαμβάνονται στο Arduino.



Εικόνα 3.13 : Flowchart του Arduino.

Κατά την ενεργοποίηση της υπολογιστικής πλατφόρμας Arduino, δημιουργούνται τα απαραίτητα αντικείμενα π.χ. GSM, GSM_SMS, EthernetServer, δηλώνονται οι μεταβλητές και ορίζονται οι ακίδες της πλατφόρμας που θα χρησιμοποιηθούν κατά την εκτέλεση του προγράμματος. Στην συνέχεια, ενεργοποιούνται οι ασπίδες GSM , Ethernet και προετοιμάζεται ο αισθητήρας κίνησης που χρησιμοποιείται στο σύστημα συναγερμού. Τέλος, εκτελείται ο βρόχος loop για όσο διάστημα η υπολογιστική πλατφόρμα παραμένει σε λειτουργία.

Αξίζει επίσης να αναφερθεί, πως η υπολογιστική πλατφόρμα λειτουργεί ως εξυπηρετητής σε αιτήσεις που δέχεται από την εφαρμογή του υπολογιστή, την οποία και αναλύουμε παρακάτω.

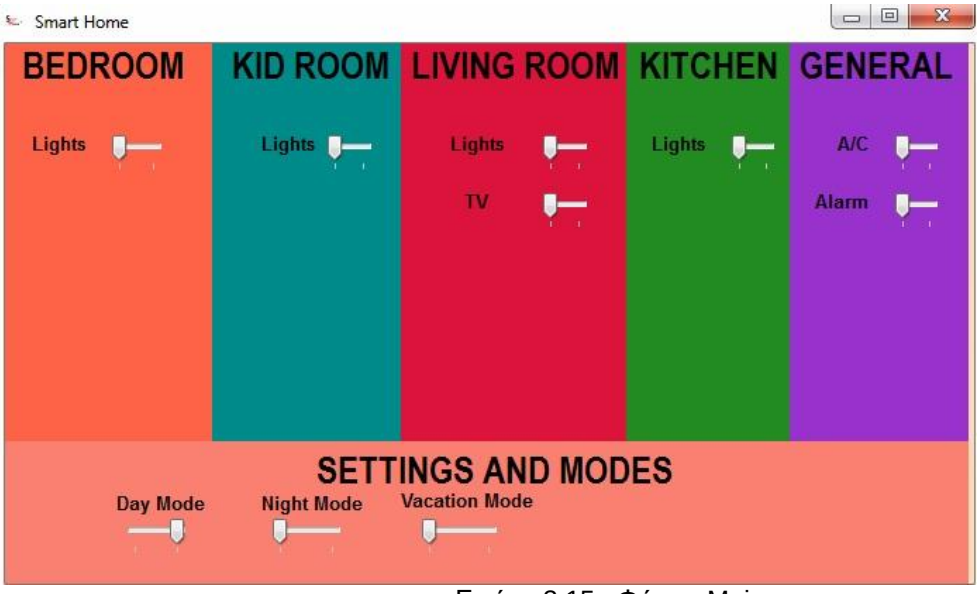
3.7. Εφαρμογή διαχείρισης του «έξυπνου» σπιτιού

Καθώς ο χρήστης επιθυμεί την διαχείριση του «έξυπνου» σπιτιού από τον υπολογιστή του, κρίθηκε απαραίτητη η δημιουργία μιας διεπαφής για το σκοπό αυτό. Η διεπαφή έχει αναπτυχθεί σε περιβάλλον οπτικού προγραμματισμού C++ (Visual C++) και αποτελείται από δύο κύριες φόρμες:

- **Φόρμα Login :** Ο χρήστης καλείται να εισάγει τους προσωπικούς του κωδικούς για την είσοδο στην εφαρμογή. Το γεγονός αυτό, κάνει το σύστημα ασφαλής από χρήστες του σπιτιού που δεν θέλουμε να έχουν πρόσβαση στην εφαρμογή.

Εικόνα 3.14 : Φόρμα Login

- **Φόρμα Main :** η οποία αποτελεί το περιβάλλον από το οποίο ο χρήστης διαχειρίζεται το «έξυπνο» σπίτι.



Εικόνα 3.15 : Φόρμα Main

Όπως παρατηρούμε κατά την ενεργοποίηση του διακόπτη προς τα δεξιά, ενεργοποιείται η συσκευή, ενώ όσο ο διακόπτης παραμένει στα αριστερά η συσκευή παραμένει κλειστή. Όταν ο χρήστης επιλέξει μια από τις καταστάσεις ημέρας ή νύχτας, η εφαρμογή ενημερώνεται αυτόματα. Κατά την ενεργοποίηση της κατάστασης διακοπών, η εφαρμογή διαχειρίζεται το σπίτι με τρόπο που έχουμε περιγράψει παραπάνω.

3.8. Σύνδεση της διεπαφής με την υπολογιστική πλατφόρμα Arduino

Η εφαρμογή επικοινωνεί με την υπολογιστική πλατφόρμα μέσω του τοπικού δίκτυο και της ασπίδας Ethernet του Arduino. Κατά την επιλογή μίας από τις διαθέσιμες λειτουργίες στο «έξυπνο» σπίτι, δημιουργείται σύνδεση με το Arduino χρησιμοποιώντας το ευρέως γνωστό πρωτόκολλο επικοινωνίας TCP/IP κατά το οποίο είναι υποχρεωτική η ρύθμιση μιας στατικής IP του δικτύου στην ασπίδα Ethernet. Στην συνέχεια αποστέλλεται στην συγκεκριμένη στατική IP ένα πακέτο δεδομένων το οποίο περιλαμβάνει την εντολή προς εκτέλεση. Η υπολογιστική πλατφόρμα Arduino λαμβάνει το παραπάνω πακέτο, και εκτελεί την κατάλληλη διαδικασία. Το πακέτο δεδομένων ονομάζεται **Socket**, ενώ η όλη διαδικασία πραγματοποιείται χρησιμοποιώντας την βιβλιοθήκη δικτυακού προγραμματισμού **WinSock2.h**. Τέλος, στο παράρτημα 2 του παρόντος συγγράμματος αναφέρεται ο κώδικας προγραμματισμού που έχει χρησιμοποιηθεί για την επίτευξη του στόχου.

Μελλοντικές επεκτάσεις της προσομοίωσης

Η προσομοίωση του «έξυπνου» σπιτιού κατασκευάστηκε με τέτοιο τρόπο ώστε η παρουσίαση της να γίνει πρακτικά και όχι θεωρητικά. Πέρα από τις τεχνολογίες και δυνατότητες που έχουν ήδη χρησιμοποιηθεί, θα μπορούσε να γίνει μελέτη για τις μελλοντικές δυνατότητες που παρουσιάζονται παρακάτω:

- Χρήση της οπτικής ίνας ως αντικατάσταση του καλωδίου Ethernet με σκοπό την επίτευξη της μέγιστης ταχύτητας στη μετάδοση δεδομένων μεταξύ της υπολογιστικής πλατφόρμας Arduino και της διεπαφής που αναπτύχθηκε.

- Χρήση ασύρματης (wireless) ασπίδας π.χ. WiFi Shield, Bluetooth Module, ως αντικατάσταση των καλωδίων Ethernet με σκοπό την ευκολότερη εγκατάσταση του συστήματος χρησιμοποιώντας πλέον ασύρματη επικοινωνία μεταξύ υπολογιστικής πλατφόρμας και διεπαφής.

- Αντικατάσταση της υπολογιστικής πλατφόρμας Arduino Uno με την πλατφόρμα Arduino Mega 2560, μιας και με τη χρήση της πρώτης, σε συνδυασμό με τις ασπίδες Ethernet και GSM, υπήρξε πρόβλημα λόγω χρήση συγκεκριμένων ακίδων από τις δύο ασπίδες που είχε ως αποτέλεσμα τον περιορισμό χρήσης των συγκεκριμένων ακίδων για κάποια από τις συσκευές της προσομοίωσης.

- Προσθήκη περισσότερων λειτουργιών και συσκευών, καθώς επίσης και περισσότερων καταστάσεων λειτουργίας, κάνοντας το σπίτι πιο αυτοματοποιημένο.

ΠΑΡΑΡΤΗΜΑΤΑ

ΠΑΡΑΡΤΗΜΑ Α. – ΚΩΔΙΚΑΣ ΤΗΣ ΥΠΟΛΟΣΤΙΚΗΣ ΠΛΑΤΦΟΡΜΑΣ ARDUINO UNO.

```
/*

 SmartHome

  Manage and control the operation of the "smart" home according
  to the commands received from user via telephone SMS or the interface
  developed in Visual C++.

The circuit.

  The design of circuit and the components that used for its implementation,
  are described in the user manual attached.

created 2015-2016
by Mitkas. D. Georgios
as thesis for A.T.E.I of Central Macedonia.
*/

/*
 Add necessary libraries.
 The SPI and Ethernet libraries are used for the connection and
communication
 between Arduino and the Visual C++ interface.
 The GSM library is used to transfer commands from the mobile phone to
 Arduino for the mentioned reasons.
 */

#include <SPI.h>
#include <Ethernet.h>
#include <GSM.h>


//Definition of the PIN code of the SIM card is in Arduino.
//The PIN code of SIM card had been deleted.

#define PINNUMBER ""
```

```
//Create GSM object for activation of the GSM Shield.

GSM gsmAccess;

//Create GSM_SMS object to receive the SMS messages.

GSM_SMS sms;

//value for saving the phone number that sent the SMS //message.

char senderNumber[20];

/*
Definition of mac, IP addresses, default gateway and subnet
mask of the network.
*/

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192, 168, 1, 71);
IPAddress gateway(192, 168, 1, 254);
IPAddress subnet(255, 255, 255, 0);



//Create Server that is listening at the given port.
//The same port is used from client application to //communicate.

EthernetServer server(27015);

//Pin orders that our components are connected to Arduino.

int alarmSensor = 5;
int buzzer = 6;
int KidroomLight = 14;
int BedroomLight = 15;
int KitchenLight = 16;
int LivingroomLight = 17;
int TV = 18;
int AC = 19;

//Definition of the seconds that the PIR sensor need to //calibrate.

int calibrationTime = 30;
```

```
//Definitions of flags that we need to activate/deactivate some actions.

boolean alarmFlag = false;
boolean kitchenFlag = false;
boolean VacationMode = false;

//Value for setting the duration that Kitchen light will be turned //on and off
//at vacation mode. The number declares milliseconds.

const int onTime = 5000;
const int offTime = 10000;


//Value for the time that Arduino is started.

unsigned long startTime;

//Value for the status of the Kitchen light.

boolean currentlyOn=false;

/*
  At setup() function, there are the declarations that they need to be
happened.
  The setup function is running only one time when we power on the Arduino.
 */



void setup()
{

   //Serial port, Ethernet Shield and GSM shield are starting now

   Serial.begin(9600);
   Ethernet.begin(mac, ip, gateway, subnet);
   gsmAccess.begin(PINNUMBER);

   //millis() function returns the number of milliseconds the //Arduino board
   began running the current program.

   startTime=millis();
```

```
  //Set the pins work like input or output at current program.

  pinMode(alarmSensor, INPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(TV, OUTPUT);
  pinMode(KidroomLight, OUTPUT);
  pinMode(BedroomLight, OUTPUT);
  pinMode(KitchenLight, OUTPUT);
  pinMode(LivingroomLight, OUTPUT);
  pinMode(AC, OUTPUT);

  //the program wait 30 seconds for PIR sensor be calibrated.

  Serial.print("calibrating sensor ");
   for(int i = 0; i < calibrationTime; i++)
     {
     Serial.print(".");

  //function that make program stops for the number of      //milliseconds are
  given.

     delay(1000);
     }

   Serial.println(" done");
   Serial.println("SENSOR ACTIVE");
   delay(50);

}
```

```
/*
 At the loop() function exists the main program of the Arduino.
 It is running while the Arduino is powered on.
 */

void loop()
{
  Serial.println("GSM initialized");
  Serial.println("Waiting for messages");

  //function that activate or deactivate the alarm system.

  checkAlarm();

  //function that activate or deactivate the vacation mode of   //home.

  checkVacationMode();

  //check for unread sms messages.

  if (sms.available())
    {
      char c;
      String req;
      Serial.println("Message received from:");

      //get remote number

      sms.remoteNumber(senderNumber, 20);

      Serial.println(senderNumber);




      // Read message bytes, save at req value and print them

      while (c = sms.read())
        {
         Serial.print(c);
         req +=c;
        }
      Serial.println("\nEND OF MESSAGE");
```

```
//act as the client ask.

  if (req == "tv on")
    digitalWrite(TV, HIGH);
  else if( req == "tv off")
    digitalWrite(TV,LOW);
  else if( req == "kidroom lights on")
    digitalWrite(KidroomLight,HIGH);
  else if( req == "kidroom lights off")
    digitalWrite(KidroomLight,LOW);
  else if( req == "bedroom lights on")
    digitalWrite(BedroomLight,HIGH);
  else if( req == "bedroom lights off")
    digitalWrite(BedroomLight,LOW);
  else if( req == "kitchen lights on")
        {
        digitalWrite(KitchenLight,HIGH);
        kitchenFlag = true;
        }
  else if( req == "kitchen lights off")
        {
        digitalWrite(KitchenLight,LOW);
        kitchenFlag = false;
        }
  else if( req == "living room lights on")
    digitalWrite(LivingroomLight,HIGH);

  else if( req == "living room lights off")
    digitalWrite(LivingroomLight,LOW);
  else if( req == "ac on")
    digitalWrite(AC, HIGH);
  else if( req == "ac off")
    digitalWrite(AC, LOW);
  else if( req == "alarm on")
    alarmFlag = true;
  else if( req == "alarm off")
    alarmFlag = false;
```

```
    else if(req == "night mode on")
       {
        digitalWrite(TV, LOW);
        digitalWrite(KidroomLight, LOW);
        digitalWrite(BedroomLight, LOW);
        digitalWrite(KitchenLight, HIGH);
        digitalWrite(LivingroomLight, LOW);
        digitalWrite(AC, LOW);
        alarmFlag = true;
        kitchenFlag = true;
       }
    else if(req == "night mode off")
       {
       digitalWrite(TV, LOW);
       digitalWrite(KidroomLight, LOW);
       digitalWrite(BedroomLight, LOW);
       digitalWrite(KitchenLight, LOW);
       digitalWrite(LivingroomLight, LOW);
       digitalWrite(AC, LOW);
       alarmFlag = false;
       kitchenFlag = false;
       }



    else if(req == "vacation mode on")
       {
       alarmFlag = true;
       VacationMode = true;
       }
    else if(req == "vacation mode off")
       {
       alarmFlag = false;
       VacationMode = false;
       }

 //delete message from SIM Card memory.

 sms.flush();

 Serial.println("MESSAGE DELETED");
}
```

```
//listen for incoming client requests.

EthernetClient client = server.available();
 if (!client)
   {
    return;
   }
Serial.println("Client connected");

//read the command and save it at string request
//then continue with execution of the request.

String request = readRequest(&client);
executeRequest(&client, request);



//close the connection between client and server

client.stop();
Serial.println("Client disonnected");

//wait 1 second befor repeat the loop function.

delay(1000);
}

void checkAlarm()
{
 //check if the alarm system is activated.

 if(alarmFlag)
   {

    //check if there is movement to the PIR Sensor
    //and if it is true , activate the alarm system.

    if(digitalRead(alarmSensor) == HIGH)
    digitalWrite(buzzer, HIGH);
```

```
    //if the movement doesnt exit , turn off the alarm system;

    else if(digitalRead(alarmSensor) == LOW)
    digitalWrite(buzzer, LOW);
   }
  else
  digitalWrite(buzzer, LOW);
}



void checkVacationMode()
{

  //check if the vacation mode at home is activated

  if(VacationMode)
  {
    //turn on the kitchen light for 5 seconds
    //and turn it off for 10 seconds after that.

    if (currentlyOn && millis()>startTime+onTime)
     {
     //Switch resistor off

     digitalWrite(KitchenLight,LOW);
     currentlyOn=false;

     //Reset timer

     startTime=millis();
     }
    if (!currentlyOn && millis()>startTime+offTime)
     {
     //Switch resistor on
     digitalWrite(KitchenLight,HIGH);
     currentlyOn=true;

     //Reset timer

     startTime=millis();
     }
  }
```

```
  //if the vacation mode of home is deactivated
  //check what was the status of the kitchen light.

  else if(!VacationMode)
  {
    if(kitchenFlag)
    digitalWrite(KitchenLight, HIGH);
    else if(!kitchenFlag)
    digitalWrite(KitchenLight, LOW);
  }
}


//read the request

String readRequest(EthernetClient* client)
{
  String  request = "";

  //loop while the client is connected.

  while (client->connected())
  {

     //read available bytes.

     while (client->available())
     {

       //read a byte.

       char c = client->read();

       //print the value

       Serial.write(c);

       //exit loop if end of line.
       if ('\n' == c)
       {
         return request;
       }
```

```
        //add byte to request line.

        request += c;
      }
  }
}

//execute the command.

void executeRequest(EthernetClient* client, String request)
{

   //check what was the command and do what it needs.

   if(request == "TVON")
    digitalWrite(TV, HIGH);
   else if(request == "TVOFF")
    digitalWrite(TV, LOW);
   else if(request == "KLightsON")
    digitalWrite(KidroomLight, HIGH);
   else if(request == "KLightsOFF")
    digitalWrite(KidroomLight, LOW);
   else if(request == "BLightsON")
      digitalWrite(BedroomLight, HIGH);

   else if(request == "BLightsOFF")
     digitalWrite(BedroomLight, LOW);
   else if(request == "KiLightsON")
       {
        digitalWrite(KitchenLight, HIGH);
        kitchenFlag = true;
       }
   else if(request == "KiLightsOFF")
       {
        digitalWrite(KitchenLight, LOW);
        kitchenFlag = false;
       }
   else if(request == "LLightsON")
    digitalWrite(LivingroomLight, HIGH);
   else if(request == "LLightsOFF")
    digitalWrite(LivingroomLight, LOW);
```

```
 else if(request == "alarmON")
  alarmFlag = true;
 else if(request == "alarmOFF")
  alarmFlag = false;
 else if(request == "ACON")
  digitalWrite(AC, HIGH);
 else if(request == "ACOFF")
  digitalWrite(AC, LOW);
 else if(request == "NightModeON")
     {
      digitalWrite(TV, LOW);
      digitalWrite(KidroomLight, LOW);
      digitalWrite(BedroomLight, LOW);
      digitalWrite(KitchenLight, HIGH);
      digitalWrite(LivingroomLight, LOW);
      digitalWrite(AC, LOW);
      alarmFlag = true;
      kitchenFlag = true;
     }

else if(request == "NightModeOFF")
     {
      digitalWrite(TV, LOW);
      digitalWrite(KidroomLight, LOW);
      digitalWrite(BedroomLight, LOW);
      digitalWrite(KitchenLight, LOW);
      digitalWrite(LivingroomLight, LOW);
      digitalWrite(AC, LOW);
      alarmFlag = false;
      kitchenFlag = false;
     }
 else if(request == "VacationModeON")
     {
     alarmFlag = true;
     VacationMode = true;
     }
 else if (request == "VacationModeOFF")


     {
     alarmFlag = false;
     VacationMode = false;
     }
```

ΠΑΡΑΡΤΗΜΑ Β. – ΚΩΔΙΚΑΣ ΔΙΕΠΑΦΗΣ SMART HOME.

ΠΑΡΑΡΤΗΜΑ Β.1 – ΚΩΔΙΚΑΣ ΦΟΡΜΑΣ LOGIN.CPP.

```
#include "Login.h"
#include "Main.h"
using namespace System;
using namespace System::Windows::Forms;
using namespace System::IO::Ports;

// these lines need to make the visual studio use Form
// Application

[STAThread]
int main(array<String^>^ arg)
{
	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);

	SmartHome::Login form;
	Application::Run(%form);

}
```

ΠΑΡΑΡΤΗΜΑ Β.2 – ΚΩΔΙΚΑΣ ΦΟΡΜΑΣ LOGIN.H.

```
#pragma once
#include "Main.h"
namespace SmartHome
{

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Summary for MyForm
	/// </summary>
	public ref class Login : public System::Windows::Forms::Form
	{
	public:
		Login(void)
		{

			InitializeComponent();
			//
			//TODO: Add the constructor code here
			//
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~Login()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::PictureBox^ pictureBox1;
	public: System::Windows::Forms::Label^  UsernameLabel;
	private:
	private: System::Windows::Forms::Label^  PasswordLabel;
	public: System::Windows::Forms::TextBox^ UsernameTextBox;



	private:
	private: System::Windows::Forms::TextBox^ PasswordTextBox;
	public:


	private: System::Windows::Forms::Button^  LoginButton;
	private: System::Windows::Forms::Button^  ExitButton;

	protected:

	private:
```

```
            /// <summary>
            /// Required designer variable.
            /// </summary>
            System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
       /// <summary>
       /// Required method for Designer support - do not modify
       /// the contents of this method with the code editor.
       /// </summary>
            void InitializeComponent(void)
            {
System::ComponentModel::ComponentResourceManager^ resources =

(gcnew System::ComponentModel::ComponentResourceManager(Login::typeid);

       this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());
       this->UsernameLabel = (gcnew System::Windows::Forms::Label());
       this->PasswordLabel = (gcnew System::Windows::Forms::Label());
       this->UsernameTextBox=(gcnew System::Windows::Forms::TextBox());
       this->PasswordTextBox=(gcnew System::Windows::Forms::TextBox());
       this->LoginButton = (gcnew System::Windows::Forms::Button());
       this->ExitButton = (gcnew System::Windows::Forms::Button());

       (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>pictureBox1))->BeginInit();
                    this->SuspendLayout();
                    //
                    // pictureBox1
                    //
                    this->pictureBox1->Image =
(cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"pictureBox1.Image")));
                    this->pictureBox1->Location = System::Drawing::Point(40,
36);
                    this->pictureBox1->Name = L"pictureBox1";
                    this->pictureBox1->Size = System::Drawing::Size(282, 183);
                    this->pictureBox1->SizeMode =
System::Windows::Forms::PictureBoxSizeMode::StretchImage;
                    this->pictureBox1->TabIndex = 0;
                    this->pictureBox1->TabStop = false;
                    //
                    // UsernameLabel
                    //
                    this->UsernameLabel->AutoSize = true;
                    this->UsernameLabel->CausesValidation = false;
                    this->UsernameLabel->Font = (gcnew
System::Drawing::Font(L"Arial", 12, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                           static_cast<System::Byte>(0)));
                    this->UsernameLabel->Location = System::Drawing::Point(53,
262);
                    this->UsernameLabel->Name = L"UsernameLabel";
                    this->UsernameLabel->Size = System::Drawing::Size(87, 19);
                    this->UsernameLabel->TabIndex = 1;
                    this->UsernameLabel->Text = L"Username";
                    this->UsernameLabel->TextAlign =
System::Drawing::ContentAlignment::MiddleCenter;
                    //
                    // PasswordLabel
                    //
                    this->PasswordLabel->AutoSize = true;
```

```
                    this->PasswordLabel->Font = (gcnew
System::Drawing::Font(L"Arial", 12, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                    this->PasswordLabel->Location = System::Drawing::Point(54,
309);
                    this->PasswordLabel->Name = L"PasswordLabel";
                    this->PasswordLabel->Size = System::Drawing::Size(86, 19);
                    this->PasswordLabel->TabIndex = 2;
                    this->PasswordLabel->Text = L"Password";
                    this->PasswordLabel->TextAlign =
System::Drawing::ContentAlignment::MiddleCenter;
                    //
                    // UsernameTextBox
                    //
                    this->UsernameTextBox->Font = (gcnew
System::Drawing::Font(L"Arial", 9, System::Drawing::FontStyle::Italic,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                    this->UsernameTextBox->Location =
System::Drawing::Point(169, 262);
                    this->UsernameTextBox->MaxLength = 16;
                    this->UsernameTextBox->Name = L"UsernameTextBox";
                    this->UsernameTextBox->Size = System::Drawing::Size(133,
21);
                    this->UsernameTextBox->TabIndex = 3;
                    this->UsernameTextBox->Text = L"Fill your username";
                    this->UsernameTextBox->TextAlign =
System::Windows::Forms::HorizontalAlignment::Center;
                    //
                    // PasswordTextBox
                    //
                    this->PasswordTextBox->Font = (gcnew
System::Drawing::Font(L"Arial", 9, System::Drawing::FontStyle::Italic,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                this->PasswordTextBox->Location = System::Drawing::Point(169,
307);
                    this->PasswordTextBox->MaxLength = 16;
                    this->PasswordTextBox->Name = L"PasswordTextBox";
                    this->PasswordTextBox->PasswordChar = '*';
                    this->PasswordTextBox->Size = System::Drawing::Size(133,
21);
                    this->PasswordTextBox->TabIndex = 4;
                    this->PasswordTextBox->TextAlign =
System::Windows::Forms::HorizontalAlignment::Center;
                    //
                    // LoginButton
                    //
                    this->LoginButton->BackColor =
System::Drawing::Color::White;
                    this->LoginButton->BackgroundImageLayout =
System::Windows::Forms::ImageLayout::Center;
                    this->LoginButton->FlatAppearance->BorderColor =
System::Drawing::Color::White;
                    this->LoginButton->FlatAppearance->MouseDownBackColor =
System::Drawing::Color::White;
```

```
                    this->LoginButton->Font = (gcnew
System::Drawing::Font(L"Arial", 12,
static_cast<System::Drawing::FontStyle>((System::Drawing::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
                            System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(0)));
                    this->LoginButton->Location = System::Drawing::Point(181,
368);
                    this->LoginButton->Name = L"LoginButton";
                    this->LoginButton->Size = System::Drawing::Size(96, 32);
                    this->LoginButton->TabIndex = 5;
                    this->LoginButton->Text = L"Login";
                    this->LoginButton->UseVisualStyleBackColor = false;
                    this->LoginButton->Click += gcnew
System::EventHandler(this, &Login::LoginButton_Click);
                    //
                    // ExitButton
                    //
                    this->ExitButton->BackColor =
System::Drawing::Color::White;
                    this->ExitButton->BackgroundImageLayout =
System::Windows::Forms::ImageLayout::Center;
                    this->ExitButton->FlatAppearance->BorderColor =
System::Drawing::Color::White;
                    this->ExitButton->FlatAppearance->MouseDownBackColor =
System::Drawing::Color::White;
                    this->ExitButton->Font = (gcnew
System::Drawing::Font(L"Arial", 12,
static_cast<System::Drawing::FontStyle>((System::Drawing::FontStyle::Bold |
System::Drawing::FontStyle::Italic)),
                            System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(0)));
                    this->ExitButton->Location = System::Drawing::Point(58,
368);
                    this->ExitButton->Name = L"ExitButton";
                    this->ExitButton->Size = System::Drawing::Size(96, 32);
                    this->ExitButton->TabIndex = 6;
                    this->ExitButton->Text = L"Exit";
                    this->ExitButton->UseVisualStyleBackColor = false;
                    //
                    // Login
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                    this->AutoSize = true;
                    this->BackColor = System::Drawing::Color::White;
                    this->ClientSize = System::Drawing::Size(334, 412);
                    this->Controls->Add(this->ExitButton);
                    this->Controls->Add(this->LoginButton);
                    this->Controls->Add(this->PasswordTextBox);
                    this->Controls->Add(this->UsernameTextBox);
                    this->Controls->Add(this->PasswordLabel);
                    this->Controls->Add(this->UsernameLabel);
                    this->Controls->Add(this->pictureBox1);
                    this->Icon =
(cli::safe_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));
                    this->Name = L"Login";
                    this->StartPosition =
System::Windows::Forms::FormStartPosition::CenterScreen;
                    this->Text = L"SmartHome";
```

```
        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>pictureBox1))->EndInit();
                        this->ResumeLayout(false);
                        this->PerformLayout();
                }
#pragma endregion
                // check if username and password are correct to continue ..
        private: System::Void LoginButton_Click(System::Object^  sender,
System::EventArgs^  e) {
                if (UsernameTextBox->Text == "1" &&  PasswordTextBox->Text ==
"1")
                {
                        Main ^MainObj = gcnew Main();
                        MainObj->Visible = true;
                        this->Hide();

                }

                else
                {
                        // if not show a message
                        MessageBox::Show("Username or Password is incorrect. Plz
try again !!");
                }
        }

};
}
```

ΠΑΡΑΡΤΗΜΑ Β.3 – ΚΩΔΙΚΑΣ ΦΟΡΜΑΣ MAIN.H.

```
#pragma once
#define WIN32_LEAN_AND_MEAN
#include <WinSock2.h>
#include <WS2tcpip.h>
#include <iostream>
#include <string>
#include <msclr/marshal_cppstd.h>
#pragma comment(lib, "Ws2_32.lib")
#pragma warning(disable:4996)
#define DEFAULT_PORT "27015"
#define DEFAULT_BUFFER_LENGTH      512

namespace SmartHome {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	using namespace System::IO::Ports;
	/// <summary>
	/// Summary for MyForm1
	/// </summary>
	public ref class Main : public System::Windows::Forms::Form
	{
	public:

		Main(void)
		{

			InitializeComponent();
			// create the socket that will connect to arduino with
commands sending.
			ConnectSocket = INVALID_SOCKET;

			// the static ip of Arduino that the program communicate.
			szServerName = "192.168.1.71";
			//
			//TODO: Add the constructor code here
			//
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~Main()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Panel^  BedRoomPanel;
	protected:
	// create the objects
	private: System::Windows::Forms::Label^  BedRoomPanelLabel;
```

```
        private: System::Windows::Forms::Label^  BedroomLightingLabel;
        private: System::Windows::Forms::TrackBar^  BedroomLightingTrackBar;
        private: System::Windows::Forms::Panel^  KidRoomPanel;
        private: System::Windows::Forms::TrackBar^  KidRoomLightingTrackBar;
        private: System::Windows::Forms::Label^  KidRoomLightingLabel;
        private: System::Windows::Forms::Label^  KidRoomPanelLabel;
        private: System::Windows::Forms::Panel^  LivingRoomPanel;
        private: System::Windows::Forms::TrackBar^  LivingRoomLightingTrackBar;
        private: System::Windows::Forms::Label^  LivingRoomLightingLabel;
        private: System::Windows::Forms::Label^  LivingRoomPanelLabel;



        private: System::Windows::Forms::Panel^  KitchenPanel;
        private: System::Windows::Forms::TrackBar^  KitchenLightingTrackBar;
        private: System::Windows::Forms::Label^  KitchenLightingLabel;
        private: System::Windows::Forms::Label^  KitchenPanelLabel;
        private: System::Windows::Forms::Panel^  GeneralPanel;
        private: System::Windows::Forms::TrackBar^  ACTrackBar;
        private: System::Windows::Forms::Label^  ACLabel;


        private: System::Windows::Forms::Label^  GeneralPanelLabel;
        private: System::ComponentModel::IContainer^  components;
        private: System::Windows::Forms::Label^  SettingsPanelLabel;
        private: System::Windows::Forms::TrackBar^  VacationModeTrackBar;
        private: System::Windows::Forms::Label^  VacationModeLabel;
        private: System::Windows::Forms::TrackBar^  NightModeTrackBar;
        private: System::Windows::Forms::Label^  NightModeLabel;
        private: System::Windows::Forms::TrackBar^  DayModeTrackBar;
        private: System::Windows::Forms::Label^  DayModeLabel;
        private: System::Windows::Forms::Panel^  SettingsAndModesPanel;


        protected:

        private:
               SOCKET ConnectSocket;
        private: System::Windows::Forms::TrackBar^  LivingRoomTVTrackBar;
        private: System::Windows::Forms::Label^  LivingRoomTVLabel;
        private: System::Windows::Forms::TrackBar^  AlarmTrackBar;
        private: System::Windows::Forms::Label^  AlarmLabel;
                      char*  szServerName;




#pragma region Windows Form Designer generated code
              /// <summary>
              /// Required method for Designer support - do not modify
              /// the contents of this method with the code editor.
              /// </summary>
              void InitializeComponent(void)
              {

                     System::ComponentModel::ComponentResourceManager^
resources = (gcnew
System::ComponentModel::ComponentResourceManager(Main::typeid));
```

```
                    this->BedRoomPanel = (gcnew
System::Windows::Forms::Panel());
                    this->BedroomLightingTrackBar = (gcnew
System::Windows::Forms::TrackBar());
                    this->BedroomLightingLabel = (gcnew
System::Windows::Forms::Label());
                    this->BedRoomPanelLabel = (gcnew
System::Windows::Forms::Label());
                    this->KidRoomPanel = (gcnew
System::Windows::Forms::Panel());
                    this->KidRoomLightingTrackBar = (gcnew
System::Windows::Forms::TrackBar());
                    this->KidRoomLightingLabel = (gcnew
System::Windows::Forms::Label());
                    this->KidRoomPanelLabel = (gcnew
System::Windows::Forms::Label());
                    this->LivingRoomPanel = (gcnew
System::Windows::Forms::Panel());
                    this->LivingRoomTVTrackBar = (gcnew
System::Windows::Forms::TrackBar());
                    this->LivingRoomTVLabel = (gcnew
System::Windows::Forms::Label());
                    this->LivingRoomLightingTrackBar = (gcnew
System::Windows::Forms::TrackBar());
                    this->LivingRoomLightingLabel = (gcnew
System::Windows::Forms::Label());
                    this->LivingRoomPanelLabel = (gcnew
System::Windows::Forms::Label());
                    this->KitchenPanel = (gcnew
System::Windows::Forms::Panel());
                    this->KitchenLightingLabel = (gcnew
System::Windows::Forms::Label());
                    this->KitchenLightingTrackBar = (gcnew
System::Windows::Forms::TrackBar());
                    this->KitchenPanelLabel = (gcnew
System::Windows::Forms::Label());
                    this->GeneralPanel = (gcnew
System::Windows::Forms::Panel());
                    this->AlarmTrackBar = (gcnew
System::Windows::Forms::TrackBar());
                    this->AlarmLabel = (gcnew
System::Windows::Forms::Label());
                    this->ACTrackBar = (gcnew
System::Windows::Forms::TrackBar());
                    this->ACLabel = (gcnew System::Windows::Forms::Label());
                    this->GeneralPanelLabel = (gcnew
System::Windows::Forms::Label());
                    this->SettingsAndModesPanel = (gcnew
System::Windows::Forms::Panel());
                    this->VacationModeTrackBar = (gcnew
System::Windows::Forms::TrackBar());
                    this->VacationModeLabel = (gcnew
System::Windows::Forms::Label());
                    this->NightModeTrackBar = (gcnew
System::Windows::Forms::TrackBar());
                    this->NightModeLabel = (gcnew
System::Windows::Forms::Label());
                    this->DayModeTrackBar = (gcnew
System::Windows::Forms::TrackBar());
                    this->DayModeLabel = (gcnew
System::Windows::Forms::Label());
```

```
                    this->SettingsPanelLabel = (gcnew
System::Windows::Forms::Label());
                    this->BedRoomPanel->SuspendLayout();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>BedroomLightingTrackBar))->BeginInit();
                    this->KidRoomPanel->SuspendLayout();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>KidRoomLightingTrackBar))->BeginInit();
                    this->LivingRoomPanel->SuspendLayout();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>LivingRoomTVTrackBar))->BeginInit();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>LivingRoomLightingTrackBar))->BeginInit();
                    this->KitchenPanel->SuspendLayout();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>KitchenLightingTrackBar))->BeginInit();
                    this->GeneralPanel->SuspendLayout();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>AlarmTrackBar))->BeginInit();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>ACTrackBar))->BeginInit();
                    this->SettingsAndModesPanel->SuspendLayout();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>VacationModeTrackBar))->BeginInit();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>NightModeTrackBar))->BeginInit();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>DayModeTrackBar))->BeginInit();
                    this->SuspendLayout();
                    //
                    // BedRoomPanel
                    //
                    this->BedRoomPanel->BackColor =
System::Drawing::Color::Tomato;
                    this->BedRoomPanel->Controls->Add(this-
>BedroomLightingTrackBar);
                    this->BedRoomPanel->Controls->Add(this-
>BedroomLightingLabel);
                    this->BedRoomPanel->Controls->Add(this-
>BedRoomPanelLabel);
                    this->BedRoomPanel->Location = System::Drawing::Point(0,
0);
                    this->BedRoomPanel->Margin =
System::Windows::Forms::Padding(0);
                    this->BedRoomPanel->Name = L"BedRoomPanel";
                    this->BedRoomPanel->Size = System::Drawing::Size(147,
282);
                    this->BedRoomPanel->TabIndex = 0;
                    //
                    // BedroomLightingTrackBar
                    //
```

```
                    this->BedroomLightingTrackBar->Location =
System::Drawing::Point(68, 63);
                    this->BedroomLightingTrackBar->Maximum = 1;
                    this->BedroomLightingTrackBar->Name =
L"BedroomLightingTrackBar";
                    this->BedroomLightingTrackBar->Size =
System::Drawing::Size(51, 45);
                    this->BedroomLightingTrackBar->TabIndex = 3;
                    this->BedroomLightingTrackBar->Scroll += gcnew
System::EventHandler(this, &Main::BedroomLightingTrackBar_Scroll);
                    this->BedroomLightingTrackBar->ValueChanged += gcnew
System::EventHandler(this, &Main::BedroomLightingTrackBar_ValueChanged);
                    //
                    // BedroomLightingLabel
                    //
                    this->BedroomLightingLabel->AutoSize = true;
                    this->BedroomLightingLabel->Font = (gcnew
System::Drawing::Font(L"Arial", 9.75F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(161)));
                    this->BedroomLightingLabel->Location =
System::Drawing::Point(16, 63);
                    this->BedroomLightingLabel->Name =
L"BedroomLightingLabel";
                    this->BedroomLightingLabel->Size =
System::Drawing::Size(46, 16);
                    this->BedroomLightingLabel->TabIndex = 2;
                    this->BedroomLightingLabel->Text = L"Lights";
                    //
                    // BedRoomPanelLabel
                    //
                    this->BedRoomPanelLabel->Anchor =
System::Windows::Forms::AnchorStyles::None;
                    this->BedRoomPanelLabel->AutoSize = true;
                    this->BedRoomPanelLabel->BackColor =
System::Drawing::Color::Tomato;
                    this->BedRoomPanelLabel->Font = (gcnew
System::Drawing::Font(L"Arial Narrow", 20, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                    this->BedRoomPanelLabel->ForeColor =
System::Drawing::SystemColors::ActiveCaptionText;
                    this->BedRoomPanelLabel->ImageAlign =
System::Drawing::ContentAlignment::TopCenter;
                    this->BedRoomPanelLabel->Location =
System::Drawing::Point(6, 0);
                    this->BedRoomPanelLabel->Name = L"BedRoomPanelLabel";
                    this->BedRoomPanelLabel->Size = System::Drawing::Size(129,
31);
                    this->BedRoomPanelLabel->TabIndex = 0;
                    this->BedRoomPanelLabel->Text = L"BEDROOM";
                    this->BedRoomPanelLabel->TextAlign =
System::Drawing::ContentAlignment::TopCenter;
                    //
                    // KidRoomPanel
                    //
                    this->KidRoomPanel->BackColor =
System::Drawing::Color::DarkCyan;
                    this->KidRoomPanel->Controls->Add(this-
>KidRoomLightingTrackBar);
                    this->KidRoomPanel->Controls->Add(this-
>KidRoomLightingLabel);
```

```
                    this->KidRoomPanel->Controls->Add(this-
>KidRoomPanelLabel);
                    this->KidRoomPanel->Location = System::Drawing::Point(147,
0);
                    this->KidRoomPanel->Margin =
System::Windows::Forms::Padding(0);
                    this->KidRoomPanel->Name = L"KidRoomPanel";
                    this->KidRoomPanel->Size = System::Drawing::Size(134,
282);
                    this->KidRoomPanel->TabIndex = 4;
                    //
                    // KidRoomLightingTrackBar
                    //
                    this->KidRoomLightingTrackBar->Location =
System::Drawing::Point(74, 63);
                    this->KidRoomLightingTrackBar->Maximum = 1;
                    this->KidRoomLightingTrackBar->Name =
L"KidRoomLightingTrackBar";
                    this->KidRoomLightingTrackBar->Size =
System::Drawing::Size(47, 45);
                    this->KidRoomLightingTrackBar->TabIndex = 3;
                    this->KidRoomLightingTrackBar->ValueChanged += gcnew
System::EventHandler(this, &Main::KidRoomLightingTrackBar_ValueChanged);
                    //
                    // KidRoomLightingLabel
                    //
                    this->KidRoomLightingLabel->AutoSize = true;
                    this->KidRoomLightingLabel->Font = (gcnew
System::Drawing::Font(L"Arial", 9.75F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(161)));
                    this->KidRoomLightingLabel->Location =
System::Drawing::Point(32, 63);
                    this->KidRoomLightingLabel->Name =
L"KidRoomLightingLabel";
                    this->KidRoomLightingLabel->Size =
System::Drawing::Size(46, 16);
                    this->KidRoomLightingLabel->TabIndex = 2;
                    this->KidRoomLightingLabel->Text = L"Lights";
                    //
                    // KidRoomPanelLabel
                    //
                    this->KidRoomPanelLabel->Anchor =
System::Windows::Forms::AnchorStyles::None;
                    this->KidRoomPanelLabel->AutoSize = true;
                    this->KidRoomPanelLabel->BackColor =
System::Drawing::Color::DarkCyan;
                    this->KidRoomPanelLabel->Font = (gcnew
System::Drawing::Font(L"Arial Narrow", 20, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                    this->KidRoomPanelLabel->ForeColor =
System::Drawing::SystemColors::ActiveCaptionText;
                    this->KidRoomPanelLabel->ImageAlign =
System::Drawing::ContentAlignment::TopCenter;
                    this->KidRoomPanelLabel->Location =
System::Drawing::Point(8, 0);
                    this->KidRoomPanelLabel->Name = L"KidRoomPanelLabel";
                    this->KidRoomPanelLabel->Size = System::Drawing::Size(126,
31);
                    this->KidRoomPanelLabel->TabIndex = 0;
                    this->KidRoomPanelLabel->Text = L"KID ROOM";
```

```
                    this->KidRoomPanelLabel->TextAlign =
System::Drawing::ContentAlignment::TopCenter;
                    //
                    // LivingRoomPanel
                    //
                    this->LivingRoomPanel->BackColor =
System::Drawing::Color::Crimson;
                    this->LivingRoomPanel->Controls->Add(this-
>LivingRoomTVTrackBar);
                    this->LivingRoomPanel->Controls->Add(this-
>LivingRoomTVLabel);
                    this->LivingRoomPanel->Controls->Add(this-
>LivingRoomLightingTrackBar);
                    this->LivingRoomPanel->Controls->Add(this-
>LivingRoomLightingLabel);
                    this->LivingRoomPanel->Controls->Add(this-
>LivingRoomPanelLabel);
                    this->LivingRoomPanel->Location =
System::Drawing::Point(281, 0);
                    this->LivingRoomPanel->Margin =
System::Windows::Forms::Padding(0);
                    this->LivingRoomPanel->Name = L"LivingRoomPanel";
                    this->LivingRoomPanel->Size = System::Drawing::Size(160,
282);
                    this->LivingRoomPanel->TabIndex = 5;
                    //
                    // LivingRoomTVTrackBar
                    //
                    this->LivingRoomTVTrackBar->Location =
System::Drawing::Point(93, 104);
                    this->LivingRoomTVTrackBar->Maximum = 1;
                    this->LivingRoomTVTrackBar->Name =
L"LivingRoomTVTrackBar";
                    this->LivingRoomTVTrackBar->Size =
System::Drawing::Size(47, 45);
                    this->LivingRoomTVTrackBar->TabIndex = 5;
                    this->LivingRoomTVTrackBar->ValueChanged += gcnew
System::EventHandler(this, &Main::LivingRoomTVTrackBar_ValueChanged);
                    //
                    // LivingRoomTVLabel
                    //
                    this->LivingRoomTVLabel->AutoSize = true;
                    this->LivingRoomTVLabel->Font = (gcnew
System::Drawing::Font(L"Arial", 9.75F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(161)));
                    this->LivingRoomTVLabel->Location =
System::Drawing::Point(42, 104);
                    this->LivingRoomTVLabel->Name = L"LivingRoomTVLabel";
                    this->LivingRoomTVLabel->Size = System::Drawing::Size(25,
16);
                    this->LivingRoomTVLabel->TabIndex = 4;
                    this->LivingRoomTVLabel->Text = L"TV";
                    //
                    // LivingRoomLightingTrackBar
                    //
                    this->LivingRoomLightingTrackBar->Location =
System::Drawing::Point(93, 63);
                    this->LivingRoomLightingTrackBar->Maximum = 1;
                    this->LivingRoomLightingTrackBar->Name =
L"LivingRoomLightingTrackBar";
```

```
                    this->LivingRoomLightingTrackBar->Size =
System::Drawing::Size(47, 45);
                    this->LivingRoomLightingTrackBar->TabIndex = 3;
                    this->LivingRoomLightingTrackBar->ValueChanged += gcnew
System::EventHandler(this, &Main::LivingRoomLightingTrackBar_ValueChanged);
                    //
                    // LivingRoomLightingLabel
                    //
                    this->LivingRoomLightingLabel->AutoSize = true;
                    this->LivingRoomLightingLabel->Font = (gcnew
System::Drawing::Font(L"Arial", 9.75F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(161)));
                    this->LivingRoomLightingLabel->Location =
System::Drawing::Point(32, 63);
                    this->LivingRoomLightingLabel->Name =
L"LivingRoomLightingLabel";
                    this->LivingRoomLightingLabel->Size =
System::Drawing::Size(46, 16);
                    this->LivingRoomLightingLabel->TabIndex = 2;
                    this->LivingRoomLightingLabel->Text = L"Lights";
                    //
                    // LivingRoomPanelLabel
                    //
                    this->LivingRoomPanelLabel->Anchor =
System::Windows::Forms::AnchorStyles::None;
                    this->LivingRoomPanelLabel->AutoSize = true;
                    this->LivingRoomPanelLabel->BackColor =
System::Drawing::Color::Crimson;
                    this->LivingRoomPanelLabel->Font = (gcnew
System::Drawing::Font(L"Arial Narrow", 20, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                    this->LivingRoomPanelLabel->ForeColor =
System::Drawing::SystemColors::ActiveCaptionText;
                    this->LivingRoomPanelLabel->ImageAlign =
System::Drawing::ContentAlignment::TopCenter;
                    this->LivingRoomPanelLabel->Location =
System::Drawing::Point(1, 0);
                    this->LivingRoomPanelLabel->Name =
L"LivingRoomPanelLabel";
                    this->LivingRoomPanelLabel->Size =
System::Drawing::Size(162, 31);
                    this->LivingRoomPanelLabel->TabIndex = 0;
                    this->LivingRoomPanelLabel->Text = L"LIVING ROOM";
                    this->LivingRoomPanelLabel->TextAlign =
System::Drawing::ContentAlignment::TopCenter;
                    //
                    // KitchenPanel
                    //
                    this->KitchenPanel->BackColor =
System::Drawing::Color::ForestGreen;
                    this->KitchenPanel->Controls->Add(this-
>KitchenLightingLabel);
                    this->KitchenPanel->Controls->Add(this-
>KitchenLightingTrackBar);
                    this->KitchenPanel->Controls->Add(this-
>KitchenPanelLabel);
                    this->KitchenPanel->Location = System::Drawing::Point(441,
0);
                    this->KitchenPanel->Margin =
System::Windows::Forms::Padding(0);
```

```
                    this->KitchenPanel->Name = L"KitchenPanel";
                    this->KitchenPanel->Size = System::Drawing::Size(116,
282);
                    this->KitchenPanel->TabIndex = 4;
                    //
                    // KitchenLightingLabel
                    //
                    this->KitchenLightingLabel->AutoSize = true;
                    this->KitchenLightingLabel->Font = (gcnew
System::Drawing::Font(L"Arial", 9.75F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(161)));
                    this->KitchenLightingLabel->Location =
System::Drawing::Point(15, 63);
                    this->KitchenLightingLabel->Name =
L"KitchenLightingLabel";
                    this->KitchenLightingLabel->Size =
System::Drawing::Size(46, 16);
                    this->KitchenLightingLabel->TabIndex = 2;
                    this->KitchenLightingLabel->Text = L"Lights";
                    //
                    // KitchenLightingTrackBar
                    //
                    this->KitchenLightingTrackBar->Location =
System::Drawing::Point(67, 63);
                    this->KitchenLightingTrackBar->Maximum = 1;
                    this->KitchenLightingTrackBar->Name =
L"KitchenLightingTrackBar";
                    this->KitchenLightingTrackBar->Size =
System::Drawing::Size(46, 45);
                    this->KitchenLightingTrackBar->TabIndex = 3;
                    this->KitchenLightingTrackBar->ValueChanged += gcnew
System::EventHandler(this, &Main::KitchenLightingTrackBar_ValueChanged);
                    //
                    // KitchenPanelLabel
                    //
                    this->KitchenPanelLabel->Anchor =
System::Windows::Forms::AnchorStyles::None;
                    this->KitchenPanelLabel->AutoSize = true;
                    this->KitchenPanelLabel->BackColor =
System::Drawing::Color::ForestGreen;
                    this->KitchenPanelLabel->Font = (gcnew
System::Drawing::Font(L"Arial Narrow", 20, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                    this->KitchenPanelLabel->ForeColor =
System::Drawing::SystemColors::ActiveCaptionText;
                    this->KitchenPanelLabel->ImageAlign =
System::Drawing::ContentAlignment::TopCenter;
                    this->KitchenPanelLabel->Location =
System::Drawing::Point(3, 0);
                    this->KitchenPanelLabel->Name = L"KitchenPanelLabel";
                    this->KitchenPanelLabel->Size = System::Drawing::Size(113,
31);
                    this->KitchenPanelLabel->TabIndex = 0;
                    this->KitchenPanelLabel->Text = L"KITCHEN";
                    this->KitchenPanelLabel->TextAlign =
System::Drawing::ContentAlignment::TopCenter;
                    //
                    // GeneralPanel
                    //
```

```
                    this->GeneralPanel->BackColor =
System::Drawing::Color::DarkOrchid;
                    this->GeneralPanel->Controls->Add(this->AlarmTrackBar);
                    this->GeneralPanel->Controls->Add(this->AlarmLabel);
                    this->GeneralPanel->Controls->Add(this->ACTrackBar);
                    this->GeneralPanel->Controls->Add(this->ACLabel);
                    this->GeneralPanel->Controls->Add(this-
>GeneralPanelLabel);
                    this->GeneralPanel->Location = System::Drawing::Point(557,
0);
                    this->GeneralPanel->Margin =
System::Windows::Forms::Padding(0);
                    this->GeneralPanel->Name = L"GeneralPanel";
                    this->GeneralPanel->Size = System::Drawing::Size(127,
282);
                    this->GeneralPanel->TabIndex = 5;
                    //
                    // AlarmTrackBar
                    //
                    this->AlarmTrackBar->Location = System::Drawing::Point(67,
104);
                    this->AlarmTrackBar->Maximum = 1;
                    this->AlarmTrackBar->Name = L"AlarmTrackBar";
                    this->AlarmTrackBar->Size = System::Drawing::Size(46, 45);
                    this->AlarmTrackBar->TabIndex = 5;
                    this->AlarmTrackBar->ValueChanged += gcnew
System::EventHandler(this, &Main::AlarmTrackBar_ValueChanged);
                    //
                    // AlarmLabel
                    //
                    this->AlarmLabel->AutoSize = true;
                    this->AlarmLabel->Font = (gcnew
System::Drawing::Font(L"Arial", 9.75F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(161)));
                    this->AlarmLabel->Location = System::Drawing::Point(15,
104);
                    this->AlarmLabel->Name = L"AlarmLabel";
                    this->AlarmLabel->Size = System::Drawing::Size(46, 16);
                    this->AlarmLabel->TabIndex = 4;
                    this->AlarmLabel->Text = L"Alarm";
                    //
                    // ACTrackBar
                    //
                    this->ACTrackBar->Location = System::Drawing::Point(67,
63);
                    this->ACTrackBar->Maximum = 1;
                    this->ACTrackBar->Name = L"ACTrackBar";
                    this->ACTrackBar->Size = System::Drawing::Size(46, 45);
                    this->ACTrackBar->TabIndex = 3;
                    this->ACTrackBar->ValueChanged += gcnew
System::EventHandler(this, &Main::ACTrackBar_ValueChanged);
                    //
                    // ACLabel
                    //
                    this->ACLabel->AutoSize = true;
                    this->ACLabel->Font = (gcnew
System::Drawing::Font(L"Arial", 9.75F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(161)));
                    this->ACLabel->Location = System::Drawing::Point(31, 63);
                    this->ACLabel->Name = L"ACLabel";
```

```
                    this->ACLabel->Size = System::Drawing::Size(30, 16);
                    this->ACLabel->TabIndex = 2;
                    this->ACLabel->Text = L"A/C";
                    //
                    // GeneralPanelLabel
                    //
                    this->GeneralPanelLabel->Anchor =
System::Windows::Forms::AnchorStyles::None;
                    this->GeneralPanelLabel->AutoSize = true;
                    this->GeneralPanelLabel->BackColor =
System::Drawing::Color::DarkOrchid;
                    this->GeneralPanelLabel->Font = (gcnew
System::Drawing::Font(L"Arial Narrow", 20, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                    this->GeneralPanelLabel->ForeColor =
System::Drawing::SystemColors::ActiveCaptionText;
                    this->GeneralPanelLabel->ImageAlign =
System::Drawing::ContentAlignment::TopCenter;
                    this->GeneralPanelLabel->Location =
System::Drawing::Point(1, 0);
                    this->GeneralPanelLabel->Name = L"GeneralPanelLabel";
                    this->GeneralPanelLabel->Size = System::Drawing::Size(123,
31);
                    this->GeneralPanelLabel->TabIndex = 0;
                    this->GeneralPanelLabel->Text = L"GENERAL";
                    this->GeneralPanelLabel->TextAlign =
System::Drawing::ContentAlignment::TopCenter;
                    //
                    // SettingsAndModesPanel
                    //
                    this->SettingsAndModesPanel->BackColor =
System::Drawing::Color::Salmon;
                    this->SettingsAndModesPanel->Controls->Add(this-
>VacationModeTrackBar);
                    this->SettingsAndModesPanel->Controls->Add(this-
>VacationModeLabel);
                    this->SettingsAndModesPanel->Controls->Add(this-
>NightModeTrackBar);
                    this->SettingsAndModesPanel->Controls->Add(this-
>NightModeLabel);
                    this->SettingsAndModesPanel->Controls->Add(this-
>DayModeTrackBar);
                    this->SettingsAndModesPanel->Controls->Add(this-
>DayModeLabel);
                    this->SettingsAndModesPanel->Controls->Add(this-
>SettingsPanelLabel);
                    this->SettingsAndModesPanel->Location =
System::Drawing::Point(0, 282);
                    this->SettingsAndModesPanel->Margin =
System::Windows::Forms::Padding(0);
                    this->SettingsAndModesPanel->Name =
L"SettingsAndModesPanel";
                    this->SettingsAndModesPanel->Size =
System::Drawing::Size(684, 101);
                    this->SettingsAndModesPanel->TabIndex = 6;
                    //
                    // VacationModeTrackBar
                    //
                    this->VacationModeTrackBar->Location =
System::Drawing::Point(288, 53);
                    this->VacationModeTrackBar->Maximum = 1;
```

```
                    this->VacationModeTrackBar->Name =
L"VacationModeTrackBar";
                    this->VacationModeTrackBar->Size =
System::Drawing::Size(69, 45);
                    this->VacationModeTrackBar->TabIndex = 7;
                    this->VacationModeTrackBar->ValueChanged += gcnew
System::EventHandler(this, &Main::VacationModeTrackBar_ValueChanged);
                    //
                    // VacationModeLabel
                    //
                    this->VacationModeLabel->AutoSize = true;
                    this->VacationModeLabel->Font = (gcnew
System::Drawing::Font(L"Arial", 9.75F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(161)));
                    this->VacationModeLabel->Location =
System::Drawing::Point(278, 34);
                    this->VacationModeLabel->Name = L"VacationModeLabel";
                    this->VacationModeLabel->Size = System::Drawing::Size(102,
16);
                    this->VacationModeLabel->TabIndex = 8;
                    this->VacationModeLabel->Text = L"Vacation Mode";
                    //
                    // NightModeTrackBar
                    //
                    this->NightModeTrackBar->Location =
System::Drawing::Point(182, 53);
                    this->NightModeTrackBar->Maximum = 1;
                    this->NightModeTrackBar->Name = L"NightModeTrackBar";
                    this->NightModeTrackBar->Size = System::Drawing::Size(64,
45);
                    this->NightModeTrackBar->TabIndex = 5;
                    this->NightModeTrackBar->ValueChanged += gcnew
System::EventHandler(this, &Main::NightModeTrackBar_ValueChanged);
                    //
                    // NightModeLabel
                    //
                    this->NightModeLabel->AutoSize = true;
                    this->NightModeLabel->Font = (gcnew
System::Drawing::Font(L"Arial", 9.75F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(161)));
                    this->NightModeLabel->Location =
System::Drawing::Point(179, 36);
                    this->NightModeLabel->Name = L"NightModeLabel";
                    this->NightModeLabel->Size = System::Drawing::Size(80,
16);
                    this->NightModeLabel->TabIndex = 6;
                    this->NightModeLabel->Text = L"Night Mode";
                    //
                    // DayModeTrackBar
                    //
                    this->DayModeTrackBar->Location =
System::Drawing::Point(79, 53);
                    this->DayModeTrackBar->Maximum = 1;
                    this->DayModeTrackBar->Name = L"DayModeTrackBar";
                    this->DayModeTrackBar->Size = System::Drawing::Size(57,
45);
                    this->DayModeTrackBar->TabIndex = 4;
                    this->DayModeTrackBar->Value = 1;
                    this->DayModeTrackBar->Scroll += gcnew
System::EventHandler(this, &Main::DayModeTrackBar_Scroll);
```

```
                    this->DayModeTrackBar->ValueChanged += gcnew
System::EventHandler(this, &Main::DayModeTrackBar_ValueChanged);
                    //
                    // DayModeLabel
                    //
                    this->DayModeLabel->AutoSize = true;
                    this->DayModeLabel->Font = (gcnew
System::Drawing::Font(L"Arial", 9.75F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(161)));
                    this->DayModeLabel->Location = System::Drawing::Point(76,
36);
                    this->DayModeLabel->Name = L"DayModeLabel";
                    this->DayModeLabel->Size = System::Drawing::Size(71, 16);
                    this->DayModeLabel->TabIndex = 4;
                    this->DayModeLabel->Text = L"Day Mode";
                    //
                    // SettingsPanelLabel
                    //
                    this->SettingsPanelLabel->Anchor =
System::Windows::Forms::AnchorStyles::None;
                    this->SettingsPanelLabel->AutoSize = true;
                    this->SettingsPanelLabel->BackColor =
System::Drawing::Color::Salmon;
                    this->SettingsPanelLabel->Font = (gcnew
System::Drawing::Font(L"Arial Narrow", 20, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                    this->SettingsPanelLabel->ForeColor =
System::Drawing::SystemColors::ActiveCaptionText;
                    this->SettingsPanelLabel->ImageAlign =
System::Drawing::ContentAlignment::TopCenter;
                    this->SettingsPanelLabel->Location =
System::Drawing::Point(215, 5);
                    this->SettingsPanelLabel->Name = L"SettingsPanelLabel";
                    this->SettingsPanelLabel->Size =
System::Drawing::Size(267, 31);
                    this->SettingsPanelLabel->TabIndex = 4;
                    this->SettingsPanelLabel->Text = L"SETTINGS AND MODES";
                    this->SettingsPanelLabel->TextAlign =
System::Drawing::ContentAlignment::TopCenter;
                    //
                    // Main
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                    this->AutoSize = true;
                    this->BackColor = System::Drawing::Color::PeachPuff;
                    this->ClientSize = System::Drawing::Size(688, 382);
                    this->Controls->Add(this->SettingsAndModesPanel);
                    this->Controls->Add(this->GeneralPanel);
                    this->Controls->Add(this->KitchenPanel);
                    this->Controls->Add(this->LivingRoomPanel);
                    this->Controls->Add(this->KidRoomPanel);
                    this->Controls->Add(this->BedRoomPanel);
                    this->FormBorderStyle =
System::Windows::Forms::FormBorderStyle::FixedDialog;
                    this->Icon =
(cli::safe_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));
                    this->Name = L"Main";
```

```
                    this->StartPosition =
System::Windows::Forms::FormStartPosition::CenterScreen;
                    this->Text = L"Smart Home";
                    this->FormClosed += gcnew
System::Windows::Forms::FormClosedEventHandler(this, &Main::Main_FormClosed_1);
                    this->BedRoomPanel->ResumeLayout(false);
                    this->BedRoomPanel->PerformLayout();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>BedroomLightingTrackBar))->EndInit();
                    this->KidRoomPanel->ResumeLayout(false);
                    this->KidRoomPanel->PerformLayout();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>KidRoomLightingTrackBar))->EndInit();
                    this->LivingRoomPanel->ResumeLayout(false);
                    this->LivingRoomPanel->PerformLayout();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>LivingRoomTVTrackBar))->EndInit();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>LivingRoomLightingTrackBar))->EndInit();
                    this->KitchenPanel->ResumeLayout(false);
                    this->KitchenPanel->PerformLayout();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>KitchenLightingTrackBar))->EndInit();
                    this->GeneralPanel->ResumeLayout(false);
                    this->GeneralPanel->PerformLayout();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>AlarmTrackBar))->EndInit();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>ACTrackBar))->EndInit();
                    this->SettingsAndModesPanel->ResumeLayout(false);
                    this->SettingsAndModesPanel->PerformLayout();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>VacationModeTrackBar))->EndInit();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>NightModeTrackBar))->EndInit();

      (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>DayModeTrackBar))->EndInit();
                    this->ResumeLayout(false);

              }
#pragma endregion
              private:

              // function for making the connection between socket and Arduino.
              void Start(void)
              {
                    WSADATA wsaData;

                    // Initialize Winsock
                    int iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);
                    if (iResult != 0)
                    {
```

```
                        MessageBox::Show("WSAStartup failed");
                        //printf("WSAStartup failed: %d\n", iResult);
                }

                struct addrinfo      *result = NULL,
                        *ptr = NULL,
                        hints;

                ZeroMemory(&hints, sizeof(hints));
                hints.ai_family = AF_UNSPEC;
                hints.ai_socktype = SOCK_STREAM;
                hints.ai_protocol = IPPROTO_TCP;

                // Resolve the server address and port
                iResult = getaddrinfo(szServerName, DEFAULT_PORT, &hints,
&result);
                if (iResult != 0)
                {
                        MessageBox::Show("getaddrinfo failed");
                        //printf("getaddrinfo failed: %d\n", iResult);
                        WSACleanup();
                }

                ptr = result;

                // Create a SOCKET for connecting to server
                ConnectSocket = socket(ptr->ai_family, ptr->ai_socktype,
ptr->ai_protocol);

                if (ConnectSocket == INVALID_SOCKET)
                {
                        MessageBox::Show("Error at socket()");
                        //printf("Error at socket(): %d\n",
WSAGetLastError());
                        freeaddrinfo(result);
                        WSACleanup();
                }

                // Connect to server
                iResult = connect(ConnectSocket, ptr->ai_addr, (int)ptr-
>ai_addrlen);

                if (iResult == SOCKET_ERROR)
                {
                        closesocket(ConnectSocket);
                        ConnectSocket = INVALID_SOCKET;
                }

                freeaddrinfo(result);

                if (ConnectSocket == INVALID_SOCKET)
                {
                        MessageBox::Show("Unable to connect to server!");
                        //printf("Unable to connect to server!\n");
                        WSACleanup();

                }


        }

        //function for sending data to Arduino using sockets.
```

```
		void Send(char* szMsg)
		{
			int iResult = send(ConnectSocket, szMsg, strlen(szMsg),
0);

			if (iResult == SOCKET_ERROR)
			{
				printf("send failed: %d\n", WSAGetLastError());
				Stop();
			}
			Stop();
		}

		//function to close the connection.
		void Stop() {
			int iResult = shutdown(ConnectSocket, SD_SEND);

			if (iResult == SOCKET_ERROR)
			{
				MessageBox::Show("shutdown failed!");
				//printf("shutdown failed: %d\n",
WSAGetLastError());
			}

			closesocket(ConnectSocket);
			WSACleanup();
		};

		//when the value of track bars is changed, the program read the
value and
		//make the connection to send the command for execution.
private: System::Void BedroomLightingTrackBar_ValueChanged(System::Object^
sender, System::EventArgs^  e) {
		if (BedroomLightingTrackBar->Value ==  1)
		{

		Start();
		Send("BLightsON");
		}
		else if (BedroomLightingTrackBar->Value == 0)
		{
		Start();
		Send("BLightsOFF");
		}

}


private: System::Void KidRoomLightingTrackBar_ValueChanged(System::Object^
sender, System::EventArgs^  e) {
	if (KidRoomLightingTrackBar->Value == 1)
	{
		Start();
		Send("KLightsON");
	}
	else if (KidRoomLightingTrackBar->Value == 0)
	{
		Start();
		Send("KLightsOFF");
	}

}
```

```
private: System::Void Main_FormClosed_1(System::Object^  sender,
System::Windows::Forms::FormClosedEventArgs^  e) {
	Application::Exit();
}
private: System::Void KitchenLightingTrackBar_ValueChanged(System::Object^
sender, System::EventArgs^  e) {
	if (KitchenLightingTrackBar->Value == 1)
	{
		Start();
		Send("KiLightsON");
	}
	else if (KitchenLightingTrackBar->Value == 0)
	{
		Start();
		Send("KiLightsOFF");
	}

}
private: System::Void ACTrackBar_ValueChanged(System::Object^  sender,
System::EventArgs^  e) {
	if (ACTrackBar->Value == 1)
	{
		Start();
		Send("ACON");
	}
	else if (ACTrackBar->Value == 0)
	{
		Start();
		Send("ACOFF");
	}

}
private: System::Void LivingRoomLightingTrackBar_ValueChanged(System::Object^
sender, System::EventArgs^  e) {
	if (LivingRoomLightingTrackBar->Value == 1)
	{
		Start();
		Send("LLightsON");
	}
	else if (LivingRoomLightingTrackBar->Value == 0)
	{
		Start();
		Send("LLightsOFF");
	}
}
private: System::Void LivingRoomTVTrackBar_ValueChanged(System::Object^
sender, System::EventArgs^  e) {
	if (LivingRoomTVTrackBar->Value == 1)
	{
		Start();
		Send("TVON");
	}
	else if (LivingRoomTVTrackBar->Value == 0)
	{
		Start();
		Send("TVOFF");
	}

}
private: System::Void AlarmTrackBar_ValueChanged(System::Object^  sender,
System::EventArgs^  e) {
```

```
        if (AlarmTrackBar->Value == 1)
        {
                Start();
                Send("alarmON");
        }
        else if (AlarmTrackBar->Value == 0)
        {
                Start();
                Send("alarmOFF");
        }
}
private: System::Void DayModeTrackBar_Scroll(System::Object^  sender,
System::EventArgs^  e) {
}
                //when the day mode is enabled , the night mode disable
automatically.
private: System::Void DayModeTrackBar_ValueChanged(System::Object^  sender,
System::EventArgs^  e) {
        if (DayModeTrackBar->Value == 1)
                NightModeTrackBar->Value = 0;
        else if (DayModeTrackBar->Value == 0)
                NightModeTrackBar->Value = 1;
}
private: System::Void NightModeTrackBar_ValueChanged(System::Object^  sender,
System::EventArgs^  e) {
        if (NightModeTrackBar->Value == 1)
        {
                DayModeTrackBar->Value = 0;
                Start();
                Send("NightModeON");
        }
        else if (NightModeTrackBar->Value == 0)
        {
                DayModeTrackBar->Value = 1;
                Start();
                Send("NightModeOFF");
        }
}
private: System::Void BedroomLightingTrackBar_Scroll(System::Object^  sender,
System::EventArgs^  e) {
}
private: System::Void VacationModeTrackBar_ValueChanged(System::Object^
sender, System::EventArgs^  e) {
        if (VacationModeTrackBar->Value == 1)
        {
                Start();
                Send("VacationModeON");
                DayModeTrackBar->Enabled = false;
                NightModeTrackBar->Enabled = false;
        }
        else if (VacationModeTrackBar->Value == 0)
        {
                Start();
                Send("VacationModeOFF");
                DayModeTrackBar->Enabled = true;
                NightModeTrackBar->Enabled = true;
        }
}
};
}
```

Βιβλιογραφία

1. Web : https://www.arduino.cc/

2. Web : https://forum.arduino.cc/

3. Web : http://playground.arduino.cc/

4. Web : https://el.wikipedia.org/wiki/Arduino

5. Web : https://en.wikipedia.org/wiki/HomePlug

6. Web : https://en.wikipedia.org/wiki/Power-line_communication

7. Web : https://msdn.microsoft.com/en-us/library/windows/desktop/ms740632(v=vs.85).aspx